Technische Universität Ilmenau
Institut für Mathematik

Preprint No. M 04/21

Zur Numerik linearer Gleichungssysteme

Vogt, Werner

November 2004

Impressum:
Hrsg.: Leiter des Instituts für Mathematik
Weimarer Straße 25
98693 Ilmenau
Tel.: +49 3677 69 3621
Fax: +49 3677 69 3270
http://www.tu-ilmenau.de/ifm/

ISSN xxxx-xxxx

ilmedia

Zur Numerik linearer Gleichungssysteme

Werner Vogt
Technische Universität Ilmenau
Institut für Mathematik
Postfach 100565
98684 Ilmenau

Ilmenau, den 1.11.2004

Zusammenfassung Der Beitrag stellt *direkte Verfahren* und *iterative Verfahren* zur numerischen Lösung linearer Gleichungssysteme vor. Die auf LU-Zerlegung und Cholesky-Zerlegung basierenden Verfahren werden analysiert und der Konditionsbegriff eingeführt. Neben den bekannten *Splitting-Verfahren* für großdimensionale Systeme werden *Krylov-Unterraum-Verfahren* wie GMRES und BiCG behandelt und anhand von Beispielen erprobt.

MSC 2000: 65F05, 65F10, 65N22

Keywords: direct methods, splitting methods, Krylov subspace methods, GMRES, BiCG

1 Direkte Verfahren für lineare Gleichungssysteme

In diesem Abschnitt werden grundlegende Methoden zur Lösung von linearen Gleichungssystemen mit n Gleichungen und n Unbekannten

$$\begin{array}{ccccccccc} a_{11}x_1 & + & a_{12}x_2 & + & \cdots & + & a_{1n}x_n & = & a_1 \\ a_{21}x_1 & + & a_{22}x_2 & + & \cdots & + & a_{2n}x_n & = & a_2 \\ \vdots & & \vdots & & & & \vdots & & \vdots \\ a_{n1}x_1 & + & a_{n2}x_2 & + & \cdots & + & a_{nn}x_n & = & a_n \end{array} \tag{1}$$

behandelt. In Matrixform lautet die Aufgabe $Ax = a$ mit der Vektor-Notation [1]

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix}, \quad a = \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix}, \quad x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}.$$

Eng damit verwandt ist das Problem, die inverse Matrix A^{-1} zu bestimmen, denn für die Lösung x gilt im Falle einer regulären Koeffizientenmatrix

$$x = A^{-1}a\,. \tag{2}$$

Auf den über- oder unterbestimmten Fall mit m Gleichungen und $n \neq m$ Unbekannten wollen wir nicht eingehen, da die Verfahren für normalbestimmte Systeme auch in diesen Fällen grundlegende Bedeutung haben. Wenn es unwesentlich ist, ob A und b reelle oder komplexe Elemente besitzen, nehmen wir vereinheitlichend an, dass $A \in \mathbb{K}^{n\times n}$ und $a \in \mathbb{K}^n$ ist, wobei $\mathbb{K} \in \{\mathbb{R}, \mathbb{C}\}$ für reelle und komplexe Systeme steht.

Aus der Linearen Algebra ist bekannt, dass das System $Ax = b$ eine eindeutige Lösung x genau dann besitzt, wenn $\det(A) \neq 0$ ist. Wir wollen deshalb voraussetzen, dass diese Bedingung erfüllt ist, womit die Darstellung (2) gerechtfertigt ist. Entscheidend für die Wahl des Lösungsverfahrens ist dann die Größe n des Systems (1): Für kleinere Systeme sind *direkte Verfahren (Eliminationsverfahren)* anwendbar. Diese liefern – bis auf Rundungsfehler durch endliche Maschinengenauigkeit – die exakte Lösung nach endlich vielen Verfahrensschritten. Bei großdimensionalen Systemen dagegen sind direkte Verfahren zu aufwändig, so dass hier *indirekte (iterative) Verfahren* vorgezogen werden, die eine Lösung nach endlich vielen Schritten nur approximativ liefern. Die Entscheidung für die konkret anzuwendende Verfahrensklasse hängt oft von der Problemstellung ab; bei voll besetzter Koeffizientenmatrix A sind direkte Verfahren bis zu $n = 2000$ durchaus praktikabel.

[1] Wir bezeichnen die rechten Seiten in der „Normalform“ $Ax = a$ stets mit a, wogegen die Bezeichnung b erst in der so genannten Fixpunktform $x = Bx + b$ benutzt wird.

1.1 LU-Zerlegung und Gauß-Algorithmus

Das aus dem Grundkurs zur Linearen Algebra bekannte Eliminationsverfahren von C.F. GAUSS[2] überführt das Gleichungssystem (1) durch geeignete Linearkombination von Zeilen in ein gestaffeltes Gleichungssystem

$$\begin{array}{rcl} r_{11}x_1 + r_{12}x_2 + \cdots + r_{1n}x_n &=& r_1 \\ r_{22}x_2 + \cdots + r_{2n}x_n &=& r_2 \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots & & \\ r_{nn}x_n &=& r_n \end{array} \quad . \tag{3}$$

Durch *Rückwärtssubstitution*, beginnend mit x_n, lässt sich die Lösung dann leicht ermitteln. Nachteilig bei dieser Vorgehensweise ist der zeitintensive Umspeicherungsaufwand von Matrixelementen pro Eliminationsschritt. Bei abgeänderten rechten Seiten a_i müsste zudem der Algorithmus komplett wiederholt werden, auch wenn sich die Koeffizientenmatrix nicht geändert hat. Deshalb ist es zweckmäßig, zuerst die Koeffizientenmatrix A mit $\mathcal{O}(n^3)$ Operationen in obere Dreiecksform zu transformieren und anschließend das System mit nur noch $\mathcal{O}(n^2)$ Operationen zu lösen.

LU-Zerlegung Wir stellen die Matrix A als Produkt zweier Dreiecksmatrizen L und R dar

$$A = L \cdot R \tag{4}$$

mit der unteren (linken) Dreiecksmatrix L und einer oberen (rechten) Dreiecksmatrix R der Form

$$L = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ l_{21} & 1 & 0 & \cdots & 0 \\ l_{31} & l_{32} & 1 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ l_{n1} & l_{n2} & l_{n3} & \cdots & 1 \end{pmatrix}, \quad R = \begin{pmatrix} r_{11} & r_{12} & r_{13} & \cdots & r_{1n} \\ 0 & r_{22} & r_{23} & \cdots & r_{2n} \\ 0 & & r_{33} & \cdots & r_{3n} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & r_{nn} \end{pmatrix}.$$

Die Darstellung (4) heisst *LR-Zerlegung oder LU-Zerlegung*[3] von A. Wegen der speziellen Hauptdiagonale ist $\det(L) = 1$ und damit die Matrix L stets regulär. Die n^2 unbekannten Koeffizienten r_{ik} und l_{ik} sind aus den n^2 Gleichungen

$$L \cdot R = A\,, \quad \text{d.h.} \qquad \sum_{j=1}^{n} l_{ij} r_{jk} = a_{ik}\,, \quad i,k = 1,2,\ldots,n$$

zu ermitteln. Diese n^2 nichtlinearen Gleichungen sind aufgrund der Dreiecksgestalt von L und R leicht lösbar. Es ergeben sich folgende einfache Vorschriften zur Ermittlung der Koeffizienten r_{ik} und der Hilfsgrößen l_{ik}:

- 1. Fall mit $i \le k$, d.h. a_{ik} liegt oberhalb oder auf der Hauptdiagonalen:

$$r_{ik} = a_{ik} - l_{i1}r_{1k} - l_{i2}r_{2k} - ... - l_{i,i-1}r_{i-1,k} \tag{5}$$

[2]Carl Friedrich Gauß (1777-1855), deutscher Mathematiker, bahnbrechend auf fast allen Gebieten der Mathematik. Anlässlich der Landesvermessung (Landestriangulation von Hannover, gemeinsam mit W. Weber) entwickelte er den nach ihm benannten Algorithmus.

[3]Im Englischen meist LU-decomposition $A = LU$ genannt, wegen der Zerlegung in eine untere (lower) und eine obere (upper) Dreiecksmatrix.

- 2. Fall mit $i > k$, d.h. a_{ik} liegt unterhalb der Hauptdiagonalen:

$$l_{ik} = (a_{ik} - l_{i1}r_{1k} - l_{i2}r_{2k} - ... - l_{i,k-1}r_{k-1,k})/r_{kk} \tag{6}$$

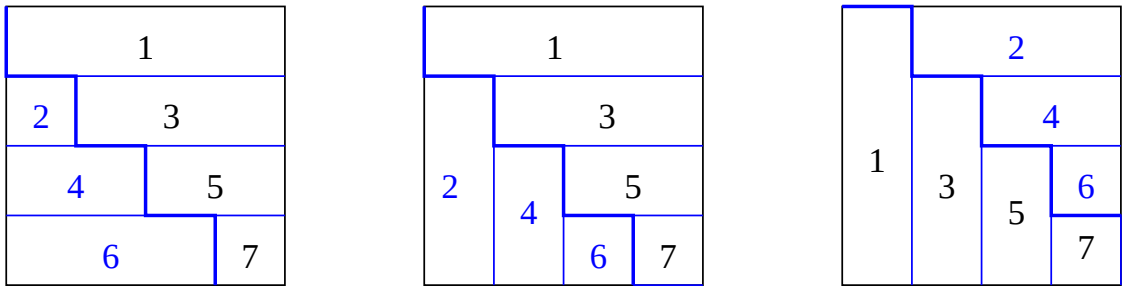
Zur Bestimmung der r_{ik} werden nur oberhalb von r_{ik} stehende Werte r_{jk} benötigt sowie links von r_{ik} stehende Werte l_{ij}. Zur Bestimmung der l_{ik} werden nur oberhalb von l_{ik} stehende Werte r_{jk} benötigt sowie links von l_{ik} stehende Werte l_{ij}. Folglich kann man die LU-Zerlegung zeilenweise von oben und links her aufbauen:[4]

Berechne für $i = 1(1)n$ die i-te Zeile:

(a) Für $k = 1(1)i-1$: l_{ik} nach (5) (7)

(b) Für $k = i(1)n$: r_{ik} nach (6)

Diese streng zeilenorientierte Darstellung in Abb. 1 geht auf BANACHIEWICZ zurück und



Parkettierung nach Banachiewicz ($l_{ii} = 1$)

Parkettierung nach Doolittle ($l_{ii} = 1$)

Parkettierung nach Crout ($r_{ii} = 1$)

Abbildung 1: 3 Parkettierungsweisen der LU-Zerlegung ($n = 4$)

ist besonders geeignet, wenn die Matrixelemente von A durch die Subdiagonalelemente von L und die Nichtnullelemente von R überspeichert werden sollen (Speicherökonomie!). Die „Parkettierung“ nach DOOLITTLE berechnet die Zeilen von R abwechselnd mit den Spalten von L, beginnend mit R nach unseren Formeln, wogegen die Variante von CROUT eine umgekehrte Reihenfolge annimmt. Der Aufwand beider Zerlegungen ist gleich.

Die *Determinantenberechnung* ergibt sich als Nebenprodukt der LU-Zerlegung mittels des Produktes der Hauptdiagonalelemente von R

$$\det(A) = \det(L) \cdot \det(R) \; = \; \prod_{i=1}^{n} r_{ii} \; . \tag{8}$$

Wir nutzen nun die LU-Zerlegung von A, um das lineare Gleichungssystem $Ax = a$ zu lösen. Einsetzen von $A = LR$ liefert

$$LRx \; = \; a \; , \quad \text{also} \quad Ly = a \; , \quad Rx = y \; ,$$

[4] Den Laufbereich eines Index werden wir anstatt mit $i = 1, 2, \ldots, n$ meist durch $i = 1(1)n$ bezeichnen. Dabei steht die Schrittweite in Klammern. Statt $i = 50, 45, 40, \ldots, 0$ schreiben wir also $i = 50(-5)0$.

womit die elementeweisen Darstellungen für die *Vorwärtssubstitution*

$$y_i = a_i - l_{i1}y_1 - l_{i,2}y_2 - \cdots - l_{i,i-1}y_{i-1}\,, \quad i = 1(1)n \tag{9}$$

und die *Rückwärtssubstitution*

$$x_i = (r_i - r_{in}x_n - r_{i,n-1}x_{n-1} - \cdots - r_{i,i+1}x_{i+1})/r_{ii}\,, \quad i = n(-1)1 \tag{10}$$

entstehen. Algorithmus 1 liefert die Lösung x und die Koeffizientendeterminante D :

Algorithmus 1 (Lineare Systeme)

Function $[x, D] =$ linearsystem (A, a)

1. LU-Zerlegung: $A = L \cdot R$ mittels (7)
2. Vorwärtssubstitution: $Ly = a$ mittels (9)
3. Rückwärtssubstitution: $Rx = y$ mittels (10)
4. Determinantenberechnung: $D = \det(A)$ mittels (8)
5. Return x und D

Beispiel 2 Wir lösen das System $Ax = a$ mit

$$A = \begin{pmatrix} 3 & 2 & -4 & 7 & -6 \\ 9 & -6 & 8 & 14 & -12 \\ -6 & 8 & -8 & -5 & 9 \\ 3 & 14 & -4 & 2 & -15 \\ 12 & -4 & 12 & 3 & 28 \end{pmatrix} \quad \text{und} \quad a = \begin{pmatrix} 6 \\ 41 \\ -19 \\ 33 \\ -67 \end{pmatrix}.$$

Die LU-Zerlegung in Schritt 1 erzeugt die Matrizen

$$L = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 3 & 1 & 0 & 0 & 0 \\ -2 & -1 & 1 & 0 & 0 \\ 1 & -1 & 5 & 1 & 0 \\ 4 & 1 & 2 & 1 & 1 \end{pmatrix}, \quad R = \begin{pmatrix} 3 & 2 & -4 & 7 & -6 \\ 0 & -12 & 20 & -7 & 6 \\ 0 & 0 & 4 & 2 & 3 \\ 0 & 0 & 0 & -22 & -18 \\ 0 & 0 & 0 & 0 & 58 \end{pmatrix},$$

die kompakt auf dem Platz von A gespeichert werden können (In der Hauptdiagonalen stehen dann die r_{ii}). Schritt 2 liefert den Hilfsvektor $y = (6, 23, 16, -30, -116)^T$, womit wir durch Rückwärtssubstitution in Schritt 3 die Lösung

$$x_1 = -5,\ x_2 = 2,\ x_3 = 4,\ x_4 = 3,\ x_5 = -2$$

sowie in Schritt 4 den Determinantenwert $\det(A) = 183744$ erhalten. □

Zeitkomplexität Wesentliche Operationen des Algorithmus 1 sind die arithmetischen Grundoperationen in $\mathbb{K}$. Für reelle Systeme benutzen Computer eine Gleitpunktarithmetik mit den Datentypen float oder double. Zeitmessungen haben ergeben, dass für eine Addition, Subtraktion oder Multiplikation etwa dieselbe Rechenzeit benötigt wird. Lediglich die Division braucht etwa die 2-3-fache Rechenzeit. Da Divisionen im Algorithmus jedoch selten auftreten (man kann mit n Divisionen auskommen), werden alle 4 arithmetischen Grundoperationen gleichgewichtet gezählt. Die LU-Zerlegung braucht ungefähr $\frac{2}{3}n^3$ Grundoperationen, wogegen die Vorwärts- und die Rückwärtssubstitution lediglich je n^2 benötigen. Der Hauptaufwand unseres Algorithmus steckt also in der LU-Zerlegung. Eine genaue Zählung ergibt die Gesamtzahl von

$$T(n) \;=\; \frac{2}{3}n^3 + \frac{3}{2}n^2 - \frac{7}{6}n \quad \approx \quad \frac{2}{3}n^3 \quad \text{für} \quad n >> 1 \tag{11}$$

Grundoperationen, so dass die Zeitkomplexität des gesamten Algorithmus $\mathcal{O}(n^3)$ lautet. Das zeigen in Abb. 2 die Rechenzeiten t_n eines typischen Anwendungsbeispiels mit Matrixeinträgen aus gleichverteilten Zufallszahlen (in MATLAB mittels `A = rand(n)`) für $n = 100, 200, \ldots, 2000$. Die Quotienten t_n/n^3 der zweiten Darstellung pegeln sich für große Dimension n auf einen konstanten Wert ein. Bei komplexwertigen Gleichungssystemen sind

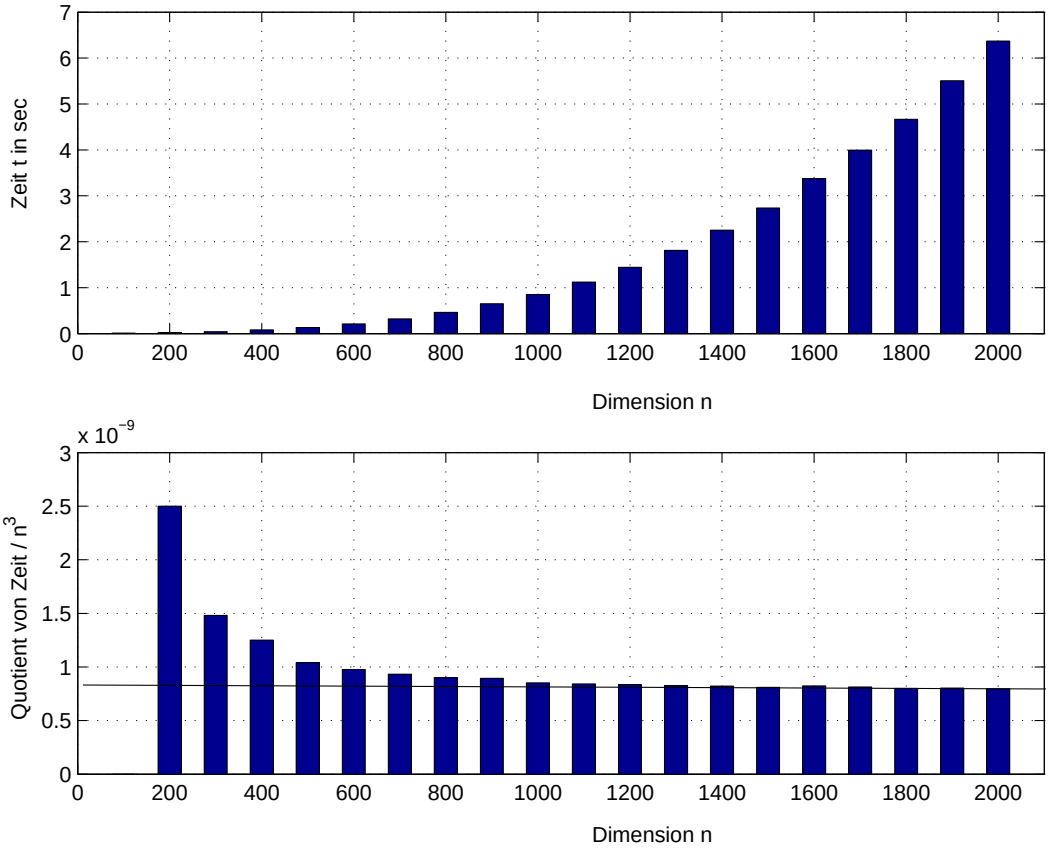


Abbildung 2: Rechenzeiten in Sekunden (Pentium 4, 1.6 GHz)

Multiplikationen allerdings zeitaufwändiger als Additionen, so dass dann der Vorfaktor von n^3 zu korrigieren ist.

Gauß-Algorithmus Im elementaren Gaußschen Algorithmus der Linearen Algebra wird das Ausgangssystem (1) ebenfalls in obere Dreiecksform transformiert. Deshalb soll nun der Zusammenhang beider Methoden gefunden werden. Dazu fassen wir die Koeffizientenmatrix und den Vektor der rechten Seiten zur erweiterten Matrix $B = (A, a)$ zusammen. Im ersten Eliminationsschritt des Gauß-Algorithmus werden die Subdiagonalelemente der 1. Spalte von B zu Null transformiert. Dies kann mittels einer *Eliminationsmatrix* L_1 notiert werden

$$B_2 = L_1 B = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ -l_{21} & 1 & 0 & \cdots & 0 \\ -l_{31} & 0 & 1 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ -l_{n1} & 0 & 0 & \cdots & 1 \end{pmatrix} \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} & a_1 \\ a_{21} & a_{22} & \cdots & a_{2n} & a_2 \\ a_{31} & a_{32} & \cdots & a_{3n} & a_3 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} & a_n \end{pmatrix}$$

$$= \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} & a_1 \\ 0 & a_{22}^{(2)} & \cdots & a_{2n}^{(2)} & a_2^{(2)} \\ 0 & a_{32}^{(2)} & \cdots & a_{3n}^{(2)} & a_3^{(2)} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & a_{n2}^{(2)} & \cdots & a_{nn}^{(2)} & a_n^{(2)} \end{pmatrix},$$

die die Multiplikatoren $l_{j1} := a_{j1}/a_{11}\,,\ j = 2(1)n$, enthält. Zur Elimination der 2. Spalte von B_2 wenden wir eine Eliminationsmatrix L_2 auf B_2 an:

$$B_3 = L_2 B_2 = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & -l_{32} & 1 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & -l_{n2} & 0 & \cdots & 1 \end{pmatrix} \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} & a_1 \\ 0 & a_{22}^{(2)} & \cdots & a_{2n}^{(2)} & a_2^{(2)} \\ 0 & a_{32}^{(2)} & \cdots & a_{3n}^{(2)} & a_3^{(2)} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & a_{n2}^{(2)} & \cdots & a_{nn}^{(2)} & a_n^{(2)} \end{pmatrix}$$

$$= \begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} & a_1 \\ 0 & a_{22}^{(2)} & a_{23}^{(2)} & \cdots & a_{2n}^{(2)} & a_2^{(2)} \\ 0 & 0 & a_{33}^{(3)} & \cdots & a_{3n}^{(3)} & a_3^{(3)} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & a_{n3}^{(3)} & \cdots & a_{nn}^{(3)} & a_n^{(3)} \end{pmatrix};$$

dabei sind die Multiplikatoren $l_{j2} := a_{j2}^{(2)}/a_{22}^{(2)}\,,\ j = 3(1)n$. Um die k-te Spalte von B_k zu eliminieren, definieren wir die k-te Eliminationsmatrix

$$L_k = \begin{pmatrix} 1 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 1 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & -l_{k+1,k} & 1 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & -l_{n,k} & 0 & \cdots & 1 \end{pmatrix} = I - l_k e_k^* \tag{12}$$

mit dem k-ten Einheitsvektor e_k in $\mathbb{K}$ und dem Eliminationsvektor

$$l_k = (0, 0, \ldots, 0, l_{k+1,k}, \ldots, l_{nk})^T\,, \quad l_{jk} := a_{jk}^{(k)}/a_{kk}^{(k)}\,.$$

Nach $n-1$ Eliminationen erhalten wir so die obere Dreiecksform des Gauß-Algorithmus

$$B_n = L_{n-1}B_{n-1} = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} & a_1 \\ 0 & a_{22}^{(2)} & a_{23}^{(2)} & \cdots & a_{2n}^{(2)} & a_2^{(2)} \\ 0 & 0 & a_{33}^{(3)} & \cdots & a_{3n}^{(3)} & a_3^{(3)} \\ & & \cdots\cdots\cdots & & & \\ 0 & 0 & 0 & \cdots & a_{nn}^{(n)} & a_n^{(n)} \end{pmatrix} = (R, r) \tag{13}$$

mit der oberen Dreiecksmatrix R und den transformierten rechten Seiten r. Sukzessives Einsetzen der ausgeführten Transformationen ergibt insgesamt die Darstellung

$$(R, r) \;=\; B_n \;=\; L_{n-1}B_{n-1} \;= \cdots = \; L_{n-1} \ldots L_2 L_1 (A, a)\;, \tag{14}$$

woraus

$$(A, a) \;=\; L_1^{-1} L_2^{-1} \cdots L_{n-1}^{-1} (R, r) \tag{15}$$

folgt. Wir bezeichnen das Produkt der inversen Eliminationsmatrizen mit L

$$L \;:=\; L_1^{-1} L_2^{-1} \cdots L_{n-1}^{-1} \tag{16}$$

und erhalten den zusammenfassenden

Satz 3 (Äquivalenz mit LU-Zerlegung) *Falls alle n Diagonalelemente $a_{kk}^{(k)}$ verschieden von Null sind, liefert der Gauß-Algorithmus gemäß (13) und (16) die Matrizen R und L der LU-Zerlegung. Der Vektor r gemäß (13) ist gleich dem Hilfsvektor y der Vorwärtssubstitution, womit $Rx = r$ mit Rückwärtssubstitution die Lösung x ergibt.*

Beweis: Nach Voraussetzung sind alle Eliminationsmatrizen L_k definiert. Die inverse Matrix hat die einfache Form $L_k^{-1} = I + l_k e_k^*$, denn

$$(I + l_k e_k^*)(I - l_k e_k^*) \;=\; I - l_k e_k^* + l_k e_k^* - l_k e_k^* l_k e_k^* \;=\; I$$

wegen $e_i^* l_k = 0$ für $i \le k$. Mit dieser Eigenschaft ergibt Einsetzen in (16)

$$\begin{aligned} L &= L_1^{-1} L_2^{-1} \cdots L_{n-1}^{-1} \\ &= (I + l_1 e_1^*)(I + l_2 e_2^*) \cdots (I + l_{n-1} e_{n-1}^*) \\ &= (I + l_1 e_1^* + l_2 e_2^*) \cdots (I + l_{n-1} e_{n-1}^*) \\ &= \cdots \\ &= I + l_1 e_1^* + l_2 e_2^* + \cdots + l_{n-1} e_{n-1}^*\,, \end{aligned}$$

also durch Ausrechnen von L und wegen (13) die Darstellungen

$$L \;=\; \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ l_{21} & 1 & 0 & \cdots & 0 \\ l_{31} & l_{32} & 1 & \cdots & 0 \\ \cdots & \cdots & \cdots & & \\ l_{n1} & l_{n2} & l_{n3} & \cdots & 1 \end{pmatrix}, \quad R \;=\; \begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ 0 & a_{22}^{(2)} & a_{23}^{(2)} & \cdots & a_{2n}^{(2)} \\ 0 & 0 & a_{33}^{(3)} & \cdots & a_{3n}^{(3)} \\ & & \cdots\cdots\cdots & & \\ 0 & 0 & 0 & \cdots & a_{nn}^{(n)} \end{pmatrix}.$$

Mit (15) ist dann $(A, a) = L(R, r)$, also $A = LR$ und $a = Lr$. □

Folgerung 4 *Die Zeitkomplexität des Gauß-Algorithmus ist $\mathcal{O}(n^3)$. In $\mathbb{R}$ benötigt der Algorithmus $\frac{2}{3}n^3$ arithmetische Gleitpunkt-Operationen.*

1.2 Pivotisierung und Pivotstrategien

Pivotisierung Im k-ten Eliminationsschritt des Gauß-Algorithmus bzw. bei der Bestimmung von l_{ik} in der LU-Zerlegung gemäß (6) ist durch das Matrixelement r_{kk} zu dividieren. Dieses zentrale Element heisst *Pivotelement*[5]. Auch bei Regularität von A kann leicht im Verlaufe der Rechnung ein Nullelement r_{kk} entstehen (z.B. kann bereits $r_{11} = 0$ sein). Was ist zu tun, wenn dieser Fall eintritt? Man vertausche die k-te Zeile mit einer p-ten Zeile ($p > k$), so dass das „neue“ $r_{kk} \neq 0$ ist. Die derart ausgewählte p-te Zeile nennt man *Pivotzeile*. Zuerst klären wir die Frage, ob mit Zeilenvertauschungen der Algorithmus immer durchführbar ist.

Satz 5 *Für eine reguläre Matrix A existiert vor dem k-ten Eliminationsschritt stets eine Zeilenvertauschung, so dass das Element $r_{kk} \neq 0$ ist.*

Beweis: Wir betrachten die Vorgehensweise des Gauß-Algorithmus, die sich als äquivalent zur LU-Zerlegung erwiesen hat. Da A regulär ist, ist $|A| \neq 0$. In der 1. Spalte von A existiert mindestens ein $a_{i1} \neq 0$, denn sonst wäre $|A| = 0$. Dieses heißt $a_{11}^{(1)}$. Nach Vertauschung der 1. mit der i-ten Zeile ergibt sich die Determinante

$$|A| = v_1 \begin{vmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{14}^{(1)} \\ a_{21}^{(1)} & a_{22}^{(1)} & \cdots & a_{2n}^{(1)} \\ \vdots & & & \\ a_{n1}^{(1)} & a_{22}^{(1)} & \cdots & a_{nn}^{(1)} \end{vmatrix},$$

wobei $v_1 = 1$ ist, falls keine Vertauschung erfolgte, ansonsten ist $v_1 = -1$. Elimination der 1. Spalte ändert den Wert $|A|$ nicht, so dass danach

$$|A| = v_1 \begin{vmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & \cdots & a_{2n}^{(2)} \\ \vdots & \vdots & & \vdots \\ 0 & a_{n2}^{(2)} & \cdots & a_{nn}^{(2)} \end{vmatrix}$$

entsteht. Im 2. Schritt betrachtet man das reduzierte System, für das wegen der Determinantenregel gilt

$$|A| = v_1 \cdot a_{11}^{(1)} \cdot \begin{vmatrix} a_{22}^{(2)} & \cdots & a_{2n}^{(2)} \\ \vdots & & \\ a_{n2}^{(2)} & \cdots & a_{nn}^{(2)} \end{vmatrix}.$$

Obige Überlegungen übertragen sich auf die reduzierten Systeme, so dass am Ende die Darstellung

$$|A| \;=\; v_1 a_{11}^{(1)} \cdot v_2 a_{22}^{(2)} \cdot \ldots \cdot v_n a_{nn}^{(n)} \;\neq\; 0$$

gilt. Also müssen tatsächlich n Pivotelemente $a_{kk}^{(k)} \neq 0$ existieren. Wegen der Äquivalenzaussage über die erhaltene obere Dreiecksmatrix ist dann $r_{kk} = a_{kk}^{(k)}$. Das Verfahren mit Pivotisierung ist somit für jede reguläre Matrix A durchführbar □

[5]Pivot [der,das]: Angel(punkt), Zapfen zum Schwenken einer ortsfesten Lafette (lat. → frz.)

Eine *Pivotstrategie* ist eine genau definierte Regel zur Bestimmung der Pivotelemente r_{kk} im k-ten Algorithmusschritt. Die so genannte *Diagonalstrategie* ohne Zeilenvertauschungen ist nur bei speziellen Matrizen, z.B. bei *diagonal dominanten* Matrizen A erfolgreich (vgl. z.B. [13]). Bei der *Spaltenmaximumstrategie(engl.: partial pivoting)* werden die Elemente r_{ik} der k-ten Spalte der R-Matrix nach (5) berechnet und anschließend eine Pivotzeile p so bestimmt, dass

$$|r_{pk}| \;=\; \max_{k\le i\le n} |r_{ik}|\,, \quad p \ge k\,, \tag{17}$$

gilt. Damit ist garantiert, dass das Pivotelement nicht verschwindet und zugleich die Elemente der Matrix L wegen (12) betragsmäßig nicht größer als 1 sind. Nimmt man eine implizite Skalierung der R-Zeilen vor, so erhält man die bessere, aber aufwändigere *relative Spaltenmaximumstrategie*, die eine Pivotzeile p mit

$$\frac{|r_{pk}|}{\sum_{j=k}^{n} |r_{pj}|} \;=\; \max_{k\le i\le n} \frac{|r_{ik}|}{\sum_{j=k}^{n} |r_{ij}|} \tag{18}$$

bestimmt. Eine nachfolgend vorzunehmende Skalierung der R-Matrix wird damit jedoch berücksichtigt. Lässt man bei der Pivotisierung auch Spaltenvertauschungen zu (die nun registriert werden müssen, da sie die Reihenfolge der Lösungskoordinaten verändern), so ist eine *Totalpivotisierung (engl.: complete pivoting)* möglich, die das Pivotelement aus dem gesamten Resttableau mittels

$$|r_{pq}| \;=\; \max_{k\le i\le n} \max_{k\le j\le n} |r_{ij}| \tag{19}$$

ermittelt. Der Suchaufwand bei der LU-Zerlegung von insgesamt $\mathcal{O}(n^3)$ Operationen ist jedoch unverhältnismäßig groß, weshalb sich diese Strategie nicht durchgesetzt hat.

Wir wollen nun Zeilenvertauschungen bei der LU-Zerlegung zulassen und sie in deren Darstellung berücksichtigen. Dazu definieren wir *Permutationsmatrizen* $P \in \mathbb{K}^{n\times n}$ aus der Einheitsmatrix I durch Vertauschen der i-ten mit der k-ten Zeile:

$$P \;=\; \begin{pmatrix} 1 & & & & & & 0 \\ & \ddots & & & & & \\ & & 0 & \cdots & 1 & & \\ & & & 1 & & & \\ & & \vdots & & \ddots & & \vdots \\ & & & & & 1 & \\ & & 1 & \cdots & 0 & & \\ & & & & & \ddots & \\ 0 & & & & & & 1 \end{pmatrix} \begin{matrix} \\ \\ \leftarrow \text{ i-te Zeile} \\ \\ \\ \\ \leftarrow \text{ k-te Zeile} \\ \\ \end{matrix}$$

Lemma 6 *P besitzt folgende leicht überprüfbare Eigenschaften:*

(i) $P = I - (e_i - e_k)(e_i - e_k)^T$ *mit dem k-ten Einheitsvektor* e_k

(ii) P *ist regulär mit* $\det(P) = 1$, *falls* $i = k$ *und* $\det(P) = -1$, *falls* $i \neq k$ *ist.*

(iii) $P^T = P$, $P^T = P^{-1}$, *also* $P^2 = I$.

(*iv*) $A := PA$ *bewirkt die Vertauschung der Zeilen* i *und* k *in Matrix* A.

(*v*) $A := AP$ *bewirkt die Vertauschung der Spalten* i *und* k *in Matrix* A.

Stellen wir die während der LU-Zerlegung vorgenommenen Zeilenvertauschungen mittels der Permutationsmatrizen $P_1, P_2, \ldots, P_{n-1}$ dar, so entsteht nun anstelle von (14) die Umformungskette

$$R \;=\; (L_{n-1}P_{n-1})\cdots(L_2P_2)(L_1P_1)A\,,$$

die sich mittels der Eigenschaften der P_i in die Anordnung

$$R \;=\; (L'_{n-1}\cdots L'_2L'_1)(P_{n-1}\cdots P_2P_1)A \tag{20}$$

überführen lässt. Die L_i sind dabei definiert durch

$$L'_i \;=\; P_{n-1}\cdots P_{i+1}L_iP_{i+1}^{-1}\cdots P_{n-1}^{-1}\,, \quad i = 1(1)n-1.$$

Umstellung von (20) liefert schließlich die LU-Zerlegung von PA

$$PA \;=\; LR \tag{21}$$

mit $P := P_{n-1}\cdots P_2P_1$ und $L := (L'_{n-1}\cdots L'_2L'_1)^{-1} = (L'_1)^{-1}(L'_2)^{-1}\cdots(L'_{n-1})^{-1}$. Die untere Dreiecksmatrix L ergibt sich dabei ganz analog zu Satz 3. In der Matrix P sind sämtliche Zeilenvertauschungen akkumuliert. In einem Computerprogramm benutzt man anstelle von P einen Permutationsvektor $\pi \in \mathbb{N}^n$, der sämtliche Permutationen registriert. Darstellung (21) führt dann auf folgende Verbesserung 7 des ursprünglichen Algorithmus 1. Die Determinantenformel (8) wird nun wegen $A = PLR$ modifiziert zu

Algorithmus 7 (Lineare Systeme mit Pivotisierung)

Function $[x, D]$ = pivotisierung (A, a)

1. LU-Zerlegung mit $PA = L \cdot R$
2. Vorwärtssubstitution: $Ly = Pa$ mittels (9)
3. Rückwärtssubstitution: $Rx = y$ mittels (10)
4. Determinantenberechnung: $D = \det(A)$ mittels (22)
5. Return x und D

$$\det(A) \;=\; \det(P)\cdot\det(L)\cdot\det(R) \;=\; (-1)^p\cdot\prod_{i=1}^{n} r_{ii}\,, \tag{22}$$

worin p die Anzahl der durchgeführten Zeilenvertauschungen ist.

Matlab Im Numerik-System MATLAB erhält man die 3 Matrizen L, R, P des Algorithmusschrittes 1 mit dem Kommando

```
[L,R,P] = lu(A)
```

zur LU-Zerlegung. Mit den Kommandos

```
y = L\(P*a)   und   x = R\y
```

oder zusammengefasst

```
x = R\(L\(P*a))
```

ergibt sich die Lösung nach den Schritten 2 und 3. Die Determinantenformel setzt man mittels

```
D = det(P)*prod(diag(R))
```

um.

Beispiel 8 1. Wir lösen nun Beispiel 2 mit Algorithmus 7. Die LU-Zerlegung `[L,R,P] = lu(A)` liefert die Matrizen

```
 L =
    1.0000         0         0         0         0
    0.2500    1.0000         0         0         0
    0.2500    0.2000    1.0000         0         0
    0.7500   -0.2000    0.4286    1.0000         0
   -0.5000    0.4000   -0.1429   -0.3333    1.0000
 R =
   12.0000   -4.0000   12.0000    3.0000   28.0000
         0   15.0000   -7.0000    1.2500  -22.0000
         0         0   -5.6000    6.0000   -8.6000
         0         0         0    9.4286  -33.7143
         0         0         0         0   19.3333
 P =
     0     0     0     0     1
     0     0     0     1     0
     1     0     0     0     0
     0     1     0     0     0
     0     0     1     0     0
```

Mittels `x = R\(L\(P*a))` ergibt sich daraus die Lösung

```
x =
   -5.0000
    2.0000
    4.0000
    3.0000
   -2.0000
```

sowie die Determinante `D = det(P)*prod(diag(R)) = 183744`.

2. Auch bei komplexwertigen linearen Systemen funktioniert Algorithmus 7 zuverlässig. Für das System $Ax = a$ mit 3 Unbekannten in $\mathbb{C}$ mit

```
A =
 -3.5628 + 2.5091i  -3.3956 + 6.4374i  -1.0713 + 1.9081i
 -9.5990 + 4.0358i  -8.0535 + 4.4207i  -2.2119 + 6.0823i
 -0.0300 + 5.3447i   0.6536 + 0.7888i  -1.3645 + 5.9997i
a =
 10.1547 - 3.3607i
  6.3018 - 4.1906i
  2.4318 - 0.0982i
```

liefert die LU-Zerlegung `[L,R,P] = lu(A)` die 3 Matrizen

```
L =
   1.0000                  0                  0
   0.2016 - 0.4720i   1.0000                  0
   0.4088 - 0.0895i  -1.0052 - 0.0788i   1.0000
R =
 -9.5990 + 4.0358i  -8.0535 + 4.4207i  -2.2119 + 6.0823i
       0             0.1904 - 3.9040i  -3.7897 + 3.7294i
       0                  0            -4.8148 + 2.6739i
P =
     0     1     0
     0     0     1
     1     0     0
```

woraus man mit `x = R\(L\(P*a))` die Lösung

```
x =
 1.66132438366803 + 1.64816095943769i
-1.06890356481200 - 1.91309641940266i
-1.46087765333357 - 1.41454225614622i
```

erhält. Das Kommando `Fehler=norm(A*x-a)` berechnet die Euklidische Norm des Residuums $Ax - a$ als ein Maß für den Fehler `Fehler=5.3326e-015` . □

Stabilität der LU-Zerlegung Um die Auswirkung von Rundungen während der LU-Zerlegung abzuschätzen, berechnen wir von jeder beteiligten Matrix $B \in \mathbb{K}^{n\times n}$ die Kennzahl

$$m(B) := \max_{i=1(1)n} \max_{k=1(1)n} |b_{ik}| \tag{23}$$

als ein Maß für die Größe der Matrixelemente (Der Wert $||B||_G = n \cdot m(B)$ heißt *Gesamtnorm* und stellt eine submultiplikative Matrixnorm gemäß Definition 19 dar). Gewinnt man die Matrizen L, R und P mittels Pivotisierung nach der Spaltenmaximumstrategie, so ist offenbar $m(P) = 1$ und $m(L) = 1$ wegen $|l_{ik}| \leq 1,\ l_{ii} = 1\ \forall\ i, k$. Falls während der Zerlegung eine signifikante Vergrößerung der vorkommenden Matrixelemente in R eintritt, so ist instabiles Verhalten zu erwarten. Wir berechnen deshalb den so genannten *Wachstumsfaktor von A* (vgl. [33])

$$\varrho := \frac{m(R)}{m(A)} = \frac{\max\limits_{i=1(1)n} \max\limits_{k=1(1)n} |r_{ik}|}{\max\limits_{i=1(1)n} \max\limits_{k=1(1)n} |a_{ik}|} . \tag{24}$$

Hat ϱ die Größenordnung 1, so ist kein signifikantes Anwachsen der Matrixelemente eingetreten und die Transformation ist numerisch stabil. Bei großem Wert ϱ kann hingegen Instabilität eintreten.

Beispiel 9 1. Das Eingangsbeispiel 2 liefert mit MATLAB nach LU-Zerlegung `[L,R,P] = lu(A)` die Kennzahlen für $m(A)$ und $m(R)$

`mA = max(max(abs(A))) = 28` und

`mR = max(max(abs(R))) = 33.7143`,

woraus sich der moderate Wachstumsfaktor $\varrho = 1.2041$ ergibt.

2. Erzeugt man für $n = 4(4)400$ mit der MATLAB-Funktion `A = randn(n)` je 20 zufällige Matrizen mit unabhängigen, normalverteilten Einträgen, so lassen sich die Wachstumsfaktoren ϱ nach Formel (24) leicht berechnen und als Punkte darstellen. In Abb. 3 ist erkennbar, dass die Werte $\varrho(n)$ von der Größenordnung $s(n) = \sqrt{n}$ (gestrichelt eingezeichnet) sind. □

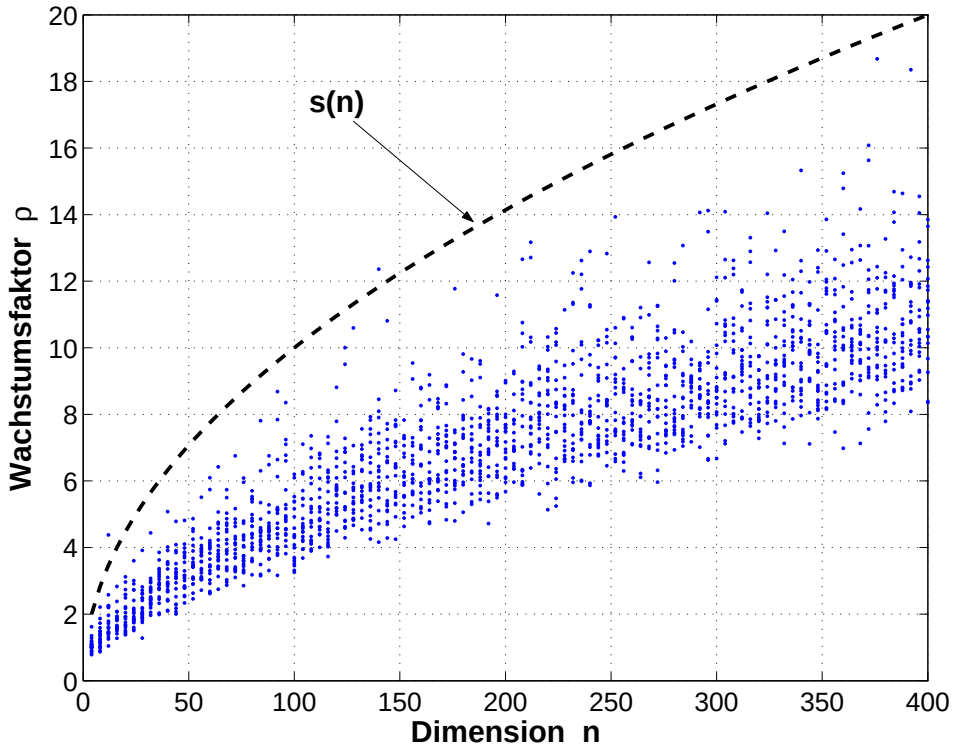


Abbildung 3: Wachstumsfaktoren für zufällig erzeugte Matrizen

1.3 Matrixinversion und Cholesky-Zerlegung

Inverse Matrizen spielen in numerischen Verfahren eine eher untergeordnete Rolle. Denn anstelle der expliziten Lösungsdarstellung $x = A^{-1}a$ mit Berechnung der Inversen von A ist es effizienter, das Gleichungssystem mittels LU-Zerlegung zu lösen. Soll dennoch $B = A^{-1}$ explizit ermittelt werden, so werden wir nachfolgend auch dafür die LU-Zerlegung benutzen. Für Matrizen A mit speziellen Eigenschaften wurden andererseits Zerlegungen entwickelt, die wesentlich schneller berechnet werden als die vorgestellte LU-Zerlegung. Wir wollen hier nur den Fall hermitescher Matrizen (bzw. symmetrischer Matrizen im reellen Fall) diskutie-

ren, die bei Diskretisierungen partieller Differenzialgleichungen in physikalischen und technischen Systemen und bei der Behandlung von Optimalitätsproblemen häufig auftreten. Weitere Matrixklassen, wie Bandmatrizen, Blockmatrizen, Toeplitz-Matrizen und unstrukturiert schwach besetzte Matrizen, werden in der Spezialliteratur [13, 25, 28, 33] behandelt.

Matrixinversion Algorithmus 7 eignet sich gut zur Berechnung der Inversen einer nichtsingulären Matrix $A \in \mathbb{K}^{n\times n}$. Nach Definition genügt die inverse Matrix $X = A^{-1}$ von A der Gleichung

$$AX = I \qquad (\text{bzw.} \quad XA = I).$$

Seien x_i die Spaltenvektoren von X und e_i die Spaltenvektoren der Einheitsmatrix I (die Einheitsvektoren), so ist die Darstellung

$$A(x_1, x_2, ..., x_n) = (e_1, e_2, ..., e_n)$$

möglich, die gleichbedeutend mit den n Gleichungssystemen

$$Ax_i = e_i\,, \quad i = 1(1)n \tag{25}$$

ist. Es sind also n lineare Gleichungssysteme mit derselben Koeffizientenmatrix und speziellen rechten Seiten zu lösen. Das kann simultan nach dem oben beschriebenen Algorithmus 7 erfolgen.

Beispiel 10 Die reelle Matrix

$$A = \begin{pmatrix} -1 & -2 & -3 & 1 \\ 2 & 7 & 3 & -2 \\ 3 & 0 & 5 & 0 \\ 4 & 2 & 8 & 1 \end{pmatrix}$$

ist zu invertieren. Die LU-Zerlegung `[L,R,P] = lu(A)` ergibt mit MATLAB die Darstellungen

```
L =
    1.0000         0         0         0
    0.5000    1.0000         0         0
    0.7500   -0.2500    1.0000         0
   -0.2500   -0.2500    1.0000    1.0000

R =
    4.0000    2.0000    8.0000    1.0000
         0    6.0000   -1.0000   -2.5000
         0         0   -1.2500   -1.3750
         0         0         0    2.0000

P =
     0     0     0     1
     0     1     0     0
     0     0     1     0
     1     0     0     0
```

woraus man nach Erzeugung der Einheitsmatrix mit dem MATLAB-Kommando `I=eye(4)` und den Vorwärts-Rückwärts-Substitutionen `X = R\(L\(P*I))` die inverse Matrix (hier nur mit 5-ziffriger Darstellung)

```
X =
    0.9167    0.3333    0.7500   -0.2500
    0.1167    0.1333   -0.2500    0.1500
   -0.5500   -0.2000   -0.2500    0.1500
    0.5000         0   -0.5000    0.5000
```

erhält. Die Defektmatrix $D = AX - I$ liefert ein Maß für die Genauigkeit des berechneten X. Sie besitzt die Zeilensummennorm `norm(A*X-I,inf) = 2.2204e-016`, die in der Größe der Maschinengenauigkeit liegt. □

Der Rechenaufwand bei der Matrizeninversion setzt sich größenordnungsmäßig aus den $\frac{2}{3}n^3$ Grundoperationen für die LU-Zerlegung und weiteren $\frac{4}{3}n^3$ Operationen für Vorwärts- und Rückwärts-Substitution zusammen. Mit dem Gesamtaufwand $T(n) \simeq 2n^3$ dauert eine Matrixinversion also etwa 3-mal so lang wie die Lösung des entsprechenden Gleichungssystems $Ax = a$ mittels LU-Zerlegung. Die Formel

$$x = A^{-1}a$$

sollte also niemals direkt in der Form `x = inv(A)*a` umgesetzt werden. Auch *simultane lineare Systeme*

$$Ax_i = a_i\,, \quad i = 1(1)m$$

mit derselben Koeffizientenmatrix A rechtfertigen nicht die Matrixinversion mittels $B = A^{-1}$. Denn der Aufwand für jede dann erforderliche Matrix-Vektor-Multiplikation $x_i = Ba_i$ ist mit $2n^2$ Operationen genauso hoch wie für jede Vorwärts- und Rückwärtssubstitution bei vorliegender LU-Zerlegung von A .

Cholesky-Zerlegung In vielen Anwendungen, z.B. in den Normalgleichungen der Quadratmittel-Approximation (engl.: least-squares approximation), treten hermitesche Koeffizientenmatrizen $A \in \mathbb{C}^{n\times n}$ mit $A^* = A$ oder symmetrische Matrizen $A \in \mathbb{R}^{n\times n}$ mit $A^T = A$ auf. In diesen Fällen nutzt man oft das Verfahren von A.L.CHOLESKY[6], das die Matrix A mittels einer speziellen LU-Zerlegung mit einer unteren Dreiecksmatrix

$$A = LL^*\,, \qquad L = \begin{pmatrix} l_{11} & 0 & \cdots & 0 \\ l_{21} & l_{22} & \cdots & 0 \\ & & \vdots & \\ l_{n1} & l_{n2} & \cdots & l_{nn} \end{pmatrix} \tag{26}$$

behandelt. Die Matrix L – und damit auch A – ist regulär, falls A hermitesch und positiv definit ist (desweiteren mit *hpd* abgekürzt).

Aus der Linearen Algebra ist bekannt, dass die Eigenwerte einer hpd-Matrix reell und positiv sind, woraus die Regularität von A folgt. Ein Definitheitsnachsweis anhand der Eigenwerte

[6] Andre-Louis Cholesky (1875-1918), französischer Offizier, befasst mit Arbeiten zur Geodäsie in Kreta und Nordafrika vor dem 1. Weltkrieg. Er entwickelte sein Verfahren zur Lösung von Normalgleichungen in der Geodäsie. Die Veröffentlichung erfolgte postum 1924 im *Bulletin Geodesique*.

oder der Hauptminoren (Sylvester'sches Determinantenkriterium) ist allerdings bei größeren Matrizen nicht praktikabel. Bei vielen Problemen ist zudem die positive Definitheit a-priori bekannt, z.B. bei linearen Ausgleichsproblemen und speziellen Klassen elliptischer Differenzialgleichungen. Hier erweist sich die Cholesky-Zerlegung als sehr nützlich.

Definition 11 (Cholesky-Zerlegung) *Die Zerlegung $A = LL^*$ mittels einer unteren Dreiecksmatrix L mit positiven Diagonalelementen l_{ii} heißt Cholesky-Zerlegung.*

Beispiel 12 Die reelle symmetrische Matrix

$$A \;=\; \begin{pmatrix} 1 & 5 & 3 & 9 \\ 5 & 26 & 16 & 47 \\ 3 & 16 & 14 & 35 \\ 9 & 47 & 35 & 95 \end{pmatrix}$$

besitzt die Cholesky-Zerlegung

$$\begin{pmatrix} 1 & 5 & 3 & 9 \\ 5 & 26 & 16 & 47 \\ 3 & 16 & 14 & 35 \\ 9 & 47 & 35 & 95 \end{pmatrix} \;=\; \begin{pmatrix} 1 & 0 & 0 & 0 \\ 5 & 1 & 0 & 0 \\ 3 & 1 & 2 & 0 \\ 9 & 2 & 3 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 5 & 3 & 9 \\ 0 & 1 & 1 & 2 \\ 0 & 0 & 2 & 3 \\ 0 & 0 & 0 & 1 \end{pmatrix} \;=\; L\,L^T\,.$$

□

In [13], S.61, wird gezeigt, dass jede hpd-Matrix eine Cholesky-Zerlegung besitzt. Die Umkehrung dieser Aussage gilt ebenfalls:

Satz 13 *Besitzt $A \in \mathbb{C}^{n\times n}$ eine Cholesky-Zerlegung, so ist A hpd-Matrix.*

Beweis: Für die hermitesch Transponierte von $A = LL^*$ folgt $A^* = (LL^*)^* = LL^* = A$, also ist A hermitesch. Mit einem beliebigen $x \in \mathbb{K}^n$ schätzt man

$$x^*Ax \;=\; x^*(LL^*)x \;=\; (L^*x)^*(L^*x) \;=\; ||L^*x||_2^2 \;\geq\; 0$$

ab. Wegen $l_{ii} > 0$ ist L^* regulär, also folgt aus $L^*x = 0$ stets $x = 0$, womit auch die positive Definitheit von A gezeigt ist. □

Damit wird sich die erfolgreiche Cholesky-Zerlegung einer hermiteschen Matrix als das effizienteste Kriterium für deren positive Definitheit erweisen. Dazu wollen wir den Algorithmus herleiten. Komponentenweise Auswertung von (26) liefert für das Element a_{ik} die Beziehung

$$\begin{aligned} a_{kk} \;&=\; \sum_{j=1}^{k} l_{kj}\overline{l_{kj}}, \qquad \text{falls } \; i = k \\ a_{ik} \;&=\; \sum_{j=1}^{k} l_{ij}\overline{l_{kj}}, \qquad \text{falls } \; i > k \end{aligned} \tag{27}$$

ist. Umstellung nach den gesuchten Elementen l_{kk} und l_{ik} ergibt die Cholesky-Zerlegung von A:

$$l_{kk} \;=\; \sqrt{a_{kk} - \sum_{j=1}^{k-1} l_{kj}\overline{l_{kj}}}\,, \qquad l_{ik} \;=\; \frac{1}{l_{kk}}\left(a_{ik} - \sum_{j=1}^{k-1} l_{ij}\overline{l_{kj}}\right)\,, \tag{28}$$

mit $i = k+1, k+2, \ldots, n,\; k = 1, 2, 3, \ldots, n$. Der Aufbau von L erfolgt offenbar spaltenweise von links nach rechts. Aus den Formeln (28) erkennen wir unmittelbar:

Folgerung 14

(*i*) *Die Cholesky-Zerlegung ist eindeutig bestimmt. Pivotisierung ist nicht erforderlich.*

(*ii*) $|l_{kj}| \leq \sqrt{a_{kk}}$ *für alle* $j,k = 1(1)n$, *d.h. die Beträge der Elemente von L sind durch die Werte* $\sqrt{a_{kk}}$ *beschränkt.*

(*iv*) *Der Determinantenwert lautet*

$$\det(A) \;=\; \left(\prod_{k=1}^{n} l_{kk}\right)^2 . \tag{29}$$

(*v*) *Die Zahl der arithmetischen Grundoperationen ist ungefähr* $\frac{1}{3}n^3$ *zuzüglich n Quadratwurzelberechnungen.*

Für größere hpd-Matrizen benötigt die Cholesky-Zerlegung deshalb nur den halben Zeitaufwand der LU-Zerlegung. Zudem garantiert sie, dass die Rundungsfehler-behaftete (reale) Zerlegung $\tilde{A} = \tilde{L}\tilde{L}^*$ trotzdem hermitesch positiv definit ist.

Ist ein lineares Gleichungssystem $Ax = a$ mit hpd-Koeffizientenmatrix A zu lösen, so setzen wir die Zerlegung (26) in das System ein

$$L\; L^* x \;=\; a$$

und definieren den Hilfsvektor $y = L^* x$. Damit erhalten wir die drei Lösungsschritte des *Cholesky-Verfahrens* 15 für lineare Gleichungssysteme. Komponentenweise lassen sich die

Algorithmus 15 (Cholesky-Verfahren)

Function $[x, D] =$ cholesky (A, a)

1. Cholesky-Zerlegung: $A = L \cdot L^*$ mittels (28)
2. Vorwärtssubstitution: $Ly = a$ mittels (30)
3. Rückwärtssubstitution: $L^* x = y$ mittels (31)
4. Determinantenberechnung: $D = \det(A)$ mittels (29)
5. Return x und D

beiden Substitutionschritte nun leicht notieren:

$$y_i \;=\; \frac{1}{l_{ii}}\left(a_i - \sum_{j=1}^{i-1} l_{ij}\, y_j\right) , \quad i = 1(1)n \tag{30}$$

$$x_i \;=\; \frac{1}{l_{ii}}\left(y_i - \sum_{k=i+1}^{n} \overline{l_{ki}}\, x_k\right) , \quad i = n(-1)1 . \tag{31}$$

Matlab Mit dem Kommando `R = chol(A)` erzeugt MATLAB die Cholesky-Zerlegung $A = R^*R$ mit der oberen Dreiecksmatrix R. Durch Transposition `L = chol(A)'` erhält man die untere Dreiecksmatrix unseres Algorithmus 15. Zusammengefasst liefert dann `x = L'\(L\a)` die Lösung des linearen Gleichungssystems.

Beispiel 16 Die symmetrische und positiv definite Pascal-Matrix der Dimension n kann man in Matlab mittel `A = pascal(n)` generieren. Sie ist von links oben her als Pascalsches Dreieck aufgebaut, d.h. $a_{ik} = a_{i,k-1} + a_{i-1,k}$. Für $n = 7$ lautet sie

```
A =
     1     1     1     1     1     1     1
     1     2     3     4     5     6     7
     1     3     6    10    15    21    28
     1     4    10    20    35    56    84
     1     5    15    35    70   126   210
     1     6    21    56   126   252   462
     1     7    28    84   210   462   924
```

Mittels `a = randperm(7)' = (6 4 7 1 2 5 3)'` erzeugen wir die rechten Seiten als Permutation der Zahlen $1, 2, \ldots, 7$ und führen die Cholesky-Zerlegung mit `L = chol(A)'` durch:

```
L =
     1     0     0     0     0     0     0
     1     1     0     0     0     0     0
     1     2     1     0     0     0     0
     1     3     3     1     0     0     0
     1     4     6     4     1     0     0
     1     5    10    10     5     1     0
     1     6    15    20    15     6     1
```

Offenbar enthält auch L das Pascalsche Dreieck. Die gesuchte Lösung x bestimmen wir mit dem Kommando `x = L'\(L\a)`

```
x = (178, -849, 1787, -2044, 1335, -471, 70)'
```

Für alle Werte $n \leq 23$ liefert das Cholesky-Verfahren mit `A = pascal(n)` trotz Rechnung in Gleitpunkt-Arithmetik die exakte Lösung x_*.

Die Cholesky-Zerlegung erweist sich als stabil bezüglich der Ausbreitung von Rundungsfehlern. Erst ab $n = 24$ wächst der relative Lösungsfehler $||x - x_*||/||x_*||$ stark an, während für $n > 30$ das Verfahren endgültig versagt (vgl. Abb. 4). Auf die Ursachen kommen wir nachfolgend zu sprechen. □

1.4 Matrixnormen, Konditionszahlen und Fehlerschätzung

Bisher wurden *Eingangsfehler* vernachlässigt, die aus ungenauen Einträgen der Koeffizientenmatrix A oder der rechten Seiten a resultieren sowie *Rechenfehler*, die wegen der endlichen *Maschinengenauigkeit* ε (im `double`-Format von MATLAB beträgt sie $2.220446 \cdot 10^{-16}$) ständig präsent sind. Während die LU-Zerlegung der 2000-reihigen Zufallsmatrix `A = rand(2000)` problemlos war (vgl. Abbildung 2), zeigt folgendes Beispiel ein gänzlich anderes Verhalten.

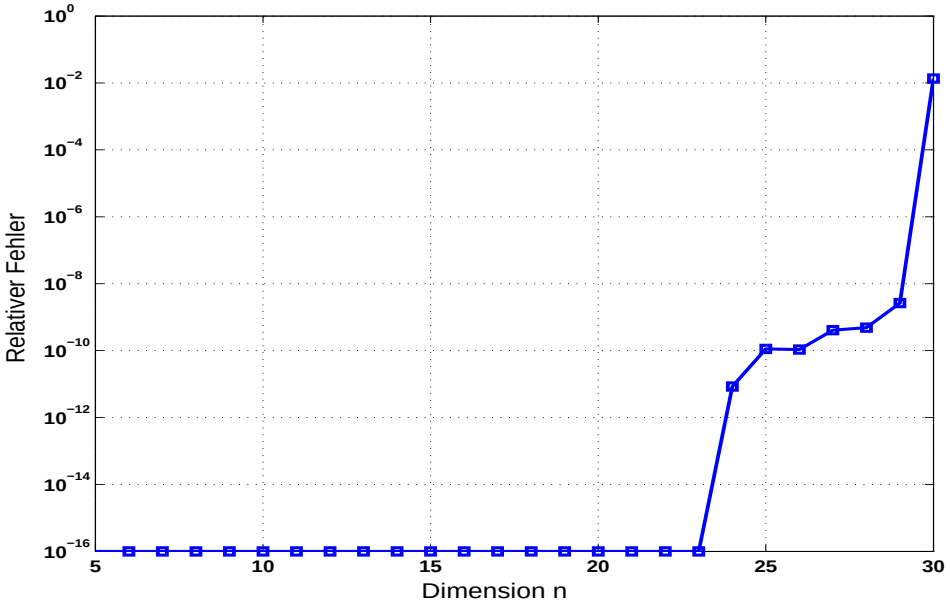


Abbildung 4: Relativer Lösungsfehler des Pascal-Systems mit $A = L\,L^T$

Beispiel 17 Wir lösen Beispiel 16 mit der Pascal-Matrix `A = pascal(n)` und den rechten Seiten `a = randperm(n)'` als Permutation der Zahlen $1, 2, \ldots, n$ mittels LU-Zerlegung `[L,R,P] = lu(A)`. Die Lösung x bestimmen wir mit dem Kommando `x= R\(L\(P*a))` und vergleichen sie mit der exakten Lösung x_*, die mittels der *Symbolic Toolbox* von MATLAB in Rationalzahlarithmetik aufwändig ermittelt wurde. Bereits ab $n = 18$ liegt der relative Lösungsfehler $||x - x_*||/||x_*||$ in der Größenordnung von 1 und signalisiert damit das Versagen des Verfahrens (vgl. Abb. 5). □

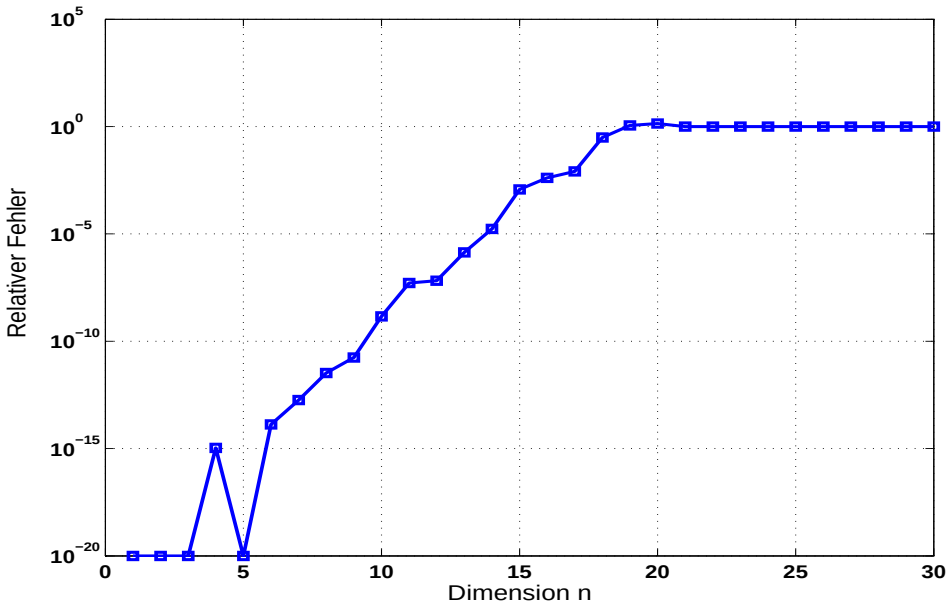


Abbildung 5: Relativer Lösungsfehler des Pascal-Systems mit $A = L\,R$

Matrixnormen Die in numerischen Fehlerdarstellungen benutzten Normen $||x||$ in $\mathbb{C}^n$ müssen leicht berechenbar sein, weshalb sich die Normen

- Betragssummennorm[7] $\quad \|x\|_1 := \sum_{j=1}^{n} |x_j|$
- Euklidische Norm $\quad \|x\|_2 := \sqrt{\sum_{j=1}^{n} |x_j|^2}$
- Betragsmaximumnorm[8] $\quad \|x\|_\infty := \max_{j=1(1)n} |x_j|$

bewährt haben. Dagegen hat die allgemeine p-Norm $\|x\|_p^p := \sum_{j=1}^{n} |x_j|^p$ geringere praktische Bedeutung. Wenn z.B. die Konvergenz einer Folge $(x_k)_{k\in\mathbb{N}}$ gegen einen Vektor x_* nachzuweisen ist, kann eine beliebige Norm gewählt werden, denn in $\mathbb{C}^n$ sind alle Normen äquivalent.

Lemma 18 (Normäquivalenz) *Alle Normen $||x||$ in $\mathbb{C}^n$ sind äquivalent in folgendem Sinne: Für jedes Paar von Normen $N_1(x)$ und $N_2(x)$ existieren Konstanten m und M, so dass $m \cdot N_2(x) \le N_1(x) \le M \cdot N_2(x) \quad \forall x \in \mathbb{C}^n$ gilt.*

Beweis: Sei speziell $N_2(x) := \|x\|_\infty = \max\limits_{i=1(1)n} |x_i|$ und $N_1(x)$ eine beliebige andere Norm. Die Menge $S = \{x \in \mathbb{C}^n | \ \|x\|_\infty = 1\}$ ist kompakt in $\mathbb{C}^n$. Wegen der Stetigkeit der Norm $N_1(x)$ existieren Konstanten

$$M := \max_{x\in S} N_1(x) > 0 \quad \text{und} \quad m := \min_{x\in S} N_1(x) > 0\,.$$

Sei $y \in \mathbb{C}^n, y \neq 0$, beliebig. Offenbar ist dann $y/\|y\|_\infty \in S$. Damit ist

$$m \le N_1\left(\frac{y}{\|y\|_\infty}\right) \le M,$$

woraus wegen der Normeigenschaft von N_1

$$m \cdot \|y\|_\infty \ \le \ N_1(y) \le M \cdot \|y\|_\infty$$

folgt. Trivialerweise gilt dies auch für $y = 0$, womit die Behauptung im Spezialfall $\|y\|_\infty = N_2(y)$ bewiesen ist. Für zwei beliebige Normen N_1, N_2 folgt nun für alle $x \in \mathbb{C}^n$

$$m_1\|x\|_\infty \le N_1(x) \le M_1 \cdot \|x\|_\infty\,,$$

$$m_2\|x\|_\infty \le N_2(x) \le M_2 \cdot \|x\|_\infty\,.$$

Man erhält damit die Abschätzungen

$$N_1(x) \le M_1 \cdot \|x\|_\infty \le \tfrac{M_1}{m_2} N_2 x = M \cdot N_2(x)\,,$$

$$N_1(x) \ge m_1 \cdot \|x\|_\infty \ge \tfrac{m_1}{M_2} N_2 x = m \cdot N_2(x)\,,$$

womit die Behauptung bewiesen wurde. □

[7] Diese Norm wird auch Manhattan-Norm genannt wegen der rechtwinkligen Straßenanordnung in Manhattan und der daraus folgenden Entfernungsangabe.

[8] Diese Norm wird auch Tschebyscheff-Norm genannt.

Mit den eingeführten Normen in $\mathbb{C}^n$ beweist man leicht

$$\begin{array}{ll} (i) & \frac{1}{\sqrt{n}}\|x\|_1 \le \|x\|_2 \le \|x\|_1 \\ (ii) & \frac{1}{n}\|x\|_1 \le \|x\|_\infty \le \|x\|_1 \\ (iii) & \frac{1}{\sqrt{n}}\|x\|_2 \le \|x\|_\infty \le \|x\|_2. \end{array} \tag{32}$$

Für quadratische Matrizen in $\mathbb{C}^{n\times n}$ führen wir nun mit den bekannten Eigenschaften der Vektornorm folgenden Normbegriff ein:

Definition 19 (Matrixnorm) *Eine Matrixnorm ist eine Abbildung von $\mathbb{C}^{n\times n}$ in $\mathbb{R}$ mit folgenden 3 Eigenschaften:*

(1) $\|A\| \ge 0\,, \quad \|A\| = 0 \Leftrightarrow A = 0\,.$

(2) $\|\lambda A\| = |\lambda|\|A\|\,, \quad \forall \lambda \in \mathbb{C}\,.$

(3) $\|A + B\| \le \|A\| + \|B\| \quad .$

Gilt weiterhin für alle $A, B \in \mathbb{C}^{n\times n}$ die Abschätzung

(4) $\|AB\| \le \|A\| \cdot \|B\| \quad ,$

so heißt die Norm submultiplikativ. Wenn eine Vektornorm $||.||$ existiert, die

(5) $||Ax|| \;\le\; ||A|| \cdot ||x|| \qquad \forall\, A \in \mathbb{C}^{n\times n},\; x \in \mathbb{C}^n$

erfüllt, so ist die Matrixnorm konsistent (verträglich, kompatibel) mit der Vektornorm.

Alle Matrixnormen sind äquivalent im Sinne des Lemmas 18; der Beweis verläuft wie im Falle der Vektornormen. Numerisch leicht auswertbare submultiplikative Matrixnormen sind die

- Zeilensummennorm (maximale absolute Zeilensumme)

$$\|A\|_\infty \;:=\; \max_{i=1(1)n} \sum_{k=1}^{n} |a_{ik}|\,, \qquad \text{konsistent mit} \quad ||x||_\infty \tag{33}$$

- Spaltensummennorm (maximale absolute Spaltensumme)

$$\|A\|_1 \;:=\; \max_{k=1(1)n} \sum_{i=1}^{n} |a_{ik}|\,, \qquad \text{konsistent mit} \quad ||x||_1 \tag{34}$$

- Frobeniusnorm

$$\|A\|_F \;:=\; \sqrt{\sum_{i=1}^{n}\sum_{k=1}^{n} |a_{ik}|^2}\,, \qquad \text{konsistent mit} \quad ||x||_2\,. \tag{35}$$

Die 4 Normeigenschaften und die Konsistenz lassen sich leicht nachweisen. Betrachten wir nun den n-dimensionalen Vektorraum $\mathbb{C}^n$ mit einer gegebenen Norm $||.||$. Durch jede $n \times n$-Matrix A wird eine lineare Abbildung von $\mathbb{C}^n$ in $\mathbb{C}^n$ der Form $y = Ax$ vermittelt.

Definition 20 (Beschränktheit) *Die lineare Abbildung $A : \mathbb{C}^n \to \mathbb{C}^n$ heißt* beschränkt, *wenn eine Konstante $C > 0$ existiert, so dass für alle $x \in \mathbb{C}^n$*

$$\|Ax\| \leq C \cdot \|x\| \qquad \textit{ist.} \tag{36}$$

Jede konsistente Matrixnorm liefert offenbar eine derartige Schranke $C = ||A||$. Die kleinste derartige Schranke ist eine charakteristische Zahl für die Matrix A.

Definition 21 (Induzierte Matrixnorm) *Die kleinste reelle Zahl C, mit der $\|Ax\| \leq C\|x\|$ für alle $x \in \mathbb{C}^n$ gilt, heißt die durch $||.||$ induzierte Norm der Matrix A und wird ebenfalls mit $||A||$ bezeichnet*[9].

Die induzierte Matrixnorm wird bei allgemeinen linearen Abbildungen (Operatoren) auch als *Operatornorm* bezeichnet. Damit lässt sie sich äquivalent formulieren:

Lemma 22

$$\|A\| \;=\; \sup_{x\neq 0} \frac{\|Ax\|}{\|x\|} \;=\; \max_{||x||=1} \|Ax\| \tag{37}$$

Beweis: Wir zeigen zuerst die linke Gleichheit. Sei $\alpha := \sup_{x\neq 0}(\|Ax\|/\|x\|)$. Offenbar ist dann $\|Ax\|/\|x\| \leq \alpha$ für alle $x \in \mathbb{C}^n$, $x \neq 0$, also $\|Ax\| \leq \alpha \|x\|$ für alle $x \in \mathbb{C}^n$. Nach Definition von $\|A\|$ ist damit $\alpha \geq \|A\|$. Aus der Supremumeigenschaft folgt, dass für beliebiges $\varepsilon > 0$ ein Element $x_\varepsilon \neq 0$ existiert, so dass $\alpha - \varepsilon \leq \|Ax_\varepsilon\|/\|x_\varepsilon\|$ gilt bzw.

$$(\alpha - \varepsilon)\|x_\varepsilon\| \leq \|Ax_\varepsilon\| \leq C\|x_\varepsilon\| \,.$$

Folglich ist $\alpha - \varepsilon \leq C$ für alle Schranken C, also $\alpha - \varepsilon \leq \inf C = \|A\|$. Da $\varepsilon > 0$ beliebig war, ist $\alpha \leq \|A\|$. Aus beiden Ungleichungen folgt die linke Gleichheit des Lemmas. Mit den Normeigenschaften erhält man schließlich

$$\sup_{y\neq 0} \frac{\|Ay\|}{\|y\|} \;=\; \sup_{y\neq 0} \|A\frac{y}{||y||}\| \;=\; \max_{||x||=1} \|Ax\| \,,$$

womit das Lemma bewiesen wurde. □

Mit Lemma 22 ist nun der Nachweis leicht zu führen, dass jede induzierte Matrixnorm tatsächlich die 4 Normeigenschaften erfüllt. Aus der Submultiplikativität folgt zudem induktiv die wichtige Abschätzung

$$||A^k|| \;\leq\; ||A||^k \quad \text{für} \quad k = 0, 1, 2, \ldots \tag{38}$$

Die Einheitsmatrix I besitzt stets die induzierte Matrixnorm $\|I\| = 1$. Die für praktische Anwendungen wesentlichsten induzierten Matrixnormen ergeben sich aus der Maximumnorm und der Betragssummennorm:

Satz 23
(*i*) *Die durch $\|x\|_\infty$ induzierte Matrixnorm ist die Zeilensummennorm (33).*
(*ii*) *Die durch $\|x\|_1$ induzierte Matrixnorm ist die Spaltensummennorm (34).*

[9] Zur Unterscheidung von anderen konsistenten Normen findet man oft die Notation $|||A|||$.

BEWEIS: Für $A = O$ (Nullmatrix) gelten die Behauptungen. Sei nun $A \neq O$.
(i) Sei $x \in \mathbb{C}^n$ und $\alpha := \max\limits_{i=1(1)n} \sum_{k=1}^{n} |a_{ik}|$. Damit schätzt man ab

$$\begin{aligned}
\|Ax\|_\infty &= \max_{i=1(1)n} \left| \sum_{k=1}^{n} a_{ik} x_k \right| \\
&\leq \max_{i=1(1)n} \sum_{k=1}^{n} |a_{ik}|\, |x_k| \;\leq\; (\max_{i=1(1)n} \sum_{k=1}^{n} |a_{ik}|) \|x\|_\infty \\
&\leq \alpha \|x\|_\infty, \quad \text{also ist } \|A\|_\infty \leq \alpha.
\end{aligned}$$

Werde $\alpha = \max\limits_{i=1(1)n} \sum\limits_{k=1(1)n} |a_{ik}|$ für einen Index $\underline{\mathrm{i}}$ erreicht, so definiert man x mit

$$x_k = \begin{cases} a_{\underline{i}k}/|a_{\underline{i}k}| & \text{für} \quad a_{\underline{i}k} \neq 0 \\ 0 & \text{für} \quad a_{\underline{i}k} = 0\,. \end{cases}$$

Offenbar ist $\|x\|_\infty = \max\limits_{k=1(1)n} |x_k| = 1$, und wir erhalten

$$\begin{aligned}
\alpha &= \max_{i=1(1)n} \sum_{k=1}^{n} |a_{ik}| \;=\; \sum_{k=1}^{n} |a_{\underline{i}k}| \;=\; \sum_{k=1}^{n} |a_{\underline{i}k} x_k| \quad \text{nach Definition von } x \\
&= |\sum_{k=1}^{n} a_{\underline{i}k} x_k| \quad \text{wegen } a_{\underline{i}k} \geq 0 \\
&\leq \max_{i=1(1)n} |\sum_{k=1}^{n} a_{ik} x_k| \;=\; \|Ax\|_\infty \;\leq\; \|A\|_\infty \|x\|_\infty \\
\alpha &\leq \|A\|_\infty,
\end{aligned}$$

also insgesamt $\|A\|_\infty = \alpha$.
(ii) Sei $x \in \mathbb{R}^n$ und $\beta := \max\limits_{k=1(1)n} \sum_{i=1}^{n} |a_{ik}|$. Abschätzung ergibt

$$\begin{aligned}
\|Ax\|_1 &= \sum_{i=1}^{n} |\sum_{k=1}^{n} a_{ik} x_k| \;\leq\; \sum_{k=1}^{n} \left(\sum_{i=1}^{n} |a_{ik}| \right) |x_k| \\
&\leq \sum_{k=1}^{n} \left(\max_{k=1(1)n} \sum_{i=1}^{n} |a_{ik}| \right) |x_k| \\
&\leq \beta \cdot \|x\|_1, \quad \text{also } \|A\|_1 \leq \beta.
\end{aligned}$$

Werde $\beta = \max\limits_{k=1(1)n} \sum\limits_{i=1(1)n} |a_{ik}|$ für einen Index $\underline{k}$ erreicht, so erhält man unter Benutzung des Einheitsvektors $e_{\underline{k}}$:

$$\begin{aligned}
\beta &= \max_{k=1(1)n} \sum_{i=1}^{n} |a_{ik}| \;=\; \sum_{i=1}^{n} |a_{i\underline{k}}| \;=\; \sum_{i=1}^{n} |\sum_{j=1}^{n} a_{ij} \cdot \delta_{j\underline{k}}| \\
&= \|A e_{\underline{k}}\|_1 \;\leq\; \|A\|_1 \|e_{\underline{k}}\|_1 \;=\; \|A_1\|\,,
\end{aligned}$$

also $\beta \leq \|A\|_1$. Insgesamt ist damit $\|A\|_1 = \beta$. □

Die durch die Euklidische Vektornorm $\|x\|_2$ induzierte Matrixnorm ist die so genannte *Spektralnorm*

$$\|A\|_2 \;:=\; \sqrt{\max_{i=1(1)n} \mu_i} \;=\; \sigma_1(A)\,, \quad \mu_i \in \sigma(A^* A)\,. \tag{39}$$

Sie erfordert die Bestimmung der Eigenwerte μ_i der hermitesch positiv semidefiniten Produktmatrix A^*A (d.h. des Gesamtspektrums $\sigma(A^*A)$ bzw. des betragsgrößten Eigenwertes) oder aber des größten Singulärwertes $\sigma_1(A)$ von A und ist deshalb für praktische Anwendungen zu aufwändig. Da oft eine Schranke für $\|A\|_2$ hinreichend ist, nutzt man die Abschätzungen

$$||A||_2 \;\le\; ||A||_F \;\le\; \sqrt{n}\,||A||_2 \qquad \text{und} \qquad ||A||_2 \;\le\; \sqrt{||A||_1||A||_\infty}\,. \tag{40}$$

Die Frobeniusnorm erfüllt zwar die 4 Normeigenschaften, sie stellt jedoch keine induzierte Matrixnorm dar. Um die untere Grenze aller Normen der Matrix A unabhängig von einer vorgegebenen Vektornorm zu ermitteln, benötigt man den Begriff des Spektralradius $\varrho(A)$ einer Matrix A , den wir hier wegen seiner Bedeutung anführen:

Definition 24 (Spektralradius) *Seien $\lambda_1, \lambda_2, \ldots, \lambda_n$ die Eigenwerte der Matrix $A \in \mathbb{C}^{n\times n}$, d.h. $\sigma(A)$ ist das Spektrum von A. Der* Spektralradius $\varrho(A)$ *ist das Maximum der Absolutbeträge aller Eigenwerte,also*

$$\varrho(A) := \max_{i=1(1)n} |\lambda_i|\,.$$

Der Spektralradius $\varrho(A)$ darf nicht mit der Spektralnorm $||A||_2$ verwechselt werden, denn $\varrho(A) = ||A||_2$ ist zwar für normale Matrizen (und damit speziell für hermitesche Matrizen) erfüllt, gilt aber nicht im Allgemeinen. Für beliebige Matrizen stellt der Spektralradius jedoch stets eine untere Schranke aller Matrixnormen dar. Dies ergibt sich aus der Eigenwertgleichung $\lambda_i v_i = Av_i$ für jeden Eigenwert λ_i mit Eigenvektor $v_i \neq 0$, woraus $|\lambda_i|||v_i|| = ||Av_i|| \le ||A||||v_i||$ folgt. Durch Maximumbildung über alle i erhält man die Schrankeneigenschaft von $\varrho(A)$. Mit größerem Aufwand lässt sich zudem nachweisen (vgl. [16]), dass der Spektralradius die untere Grenze aller induzierten Matrixnormen bildet:

Satz 25 *Bei gegebener Matrix $A \in \mathbb{C}^{n\times n}$ existiert zu jedem $\varepsilon > 0$ eine induzierte Matrixnorm $||A||$, so dass $\varrho(A) \le ||A|| \le \varrho(A) + \varepsilon$ gilt.*

Beispiel 26 Für die symmetrisch positiv definite Pascal-Matrix

```
A = pascal(6) =
     1     1     1     1     1     1
     1     2     3     4     5     6
     1     3     6    10    15    21
     1     4    10    20    35    56
     1     5    15    35    70   126
     1     6    21    56   126   252
```

liefern die MATLAB-Kommandos `norm(A,p)` folgende Normen:

- Spektralnorm $||A||_2$: `norm(A) = norm(A,2) = 332.8463`
- Spaltensummennorm $||A||_1$: `norm(A,1) = 462`
- Zeilensummennorm $||A||_\infty$: `norm(A,inf) = 462`
- Frobeniusnorm $||A||_F$: `norm(A,'fro') = 333.2161`

Mit der MATLAB-Anweisung `rho=max(abs(eig(A)))` berechnen wir vergleichsweise den Spektralradius $\varrho(A) = 332.8463$, der mit der Spektralnorm übereinstimmt. □

Konditionszahlen Im Ergebnis eines direkten Verfahrens, angewandt auf $Ax = a$, erhält man i.A. nur eine Näherung $\tilde{x}$ für die exakte Lösung x. Wesentliche Ursachen sind die *Eingangsfehler* von A und a durch ungenaue Daten und deren ungenaue Darstellung im Computer sowie *Rechenfehler* auf Grund ungenauer Rechnung – auch bei exakt vorgegebenen Daten. Wir wollen nachfolgend nur die Fortpflanzung der Eingangsfehler ΔA und Δa auf den Ergebnisfehler Δx untersuchen. Die schwierigere Aufgabe der Rundungsfehleranalyse eines konkreten Verfahrens wird in der Literatur, z.B. in [16] behandelt. Die so genannte „Vorwärtsanalyse“ schätzt den Gesamtfehler $\tilde{x} - x$ des Ergebnisses ab, wenn die Eingangsgrößen A und a mit Störungen ΔA und Δa behaftet sind. Anstelle der exakten Lösung x von $Ax = a$ erhalten wir dann eine gestörte Lösung $\tilde{x} = x + \Delta x$ des Gleichungssystems

$$\tilde{A}\tilde{x} \;=\; \tilde{a}, \qquad \text{mit} \quad \tilde{A} = A + \Delta A, \;\; \tilde{a} = a + \Delta a, \;\; \tilde{x} = x + \Delta x\,. \tag{41}$$

Gesucht ist eine Abschätzung für den *relativen Fehler* $\|\Delta x\|/\|x\|$, die vor der Rechnung angegeben werden kann (daher auch „a-priori-Schätzung“). Dazu wollen wir vorerst annehmen, dass die Matrixelemente A exakt gegeben sind, also $\Delta A = 0$ gesetzt wird. Die Frage, wie sich dann der Fehler der Eingangsgrößen $\|\Delta a\|/\|a\|$ auf den Fehler der Lösung $\|\Delta x\|/\|x\|$ auswirkt, wird vermittels der Konditionszahl $cond(A)$ der Matrix A beantwortet.

Definition 27 (Konditionszahl) *Zu gegebenem regulären $A \in \mathbb{C}^{n\times n}$ und induzierter Matrixnorm $\|A\|$ heißt*

$$cond(A) \;:=\; \|A\| \cdot \|A^{-1}\|$$

(relative) Konditionszahl *von A bezüglich der Norm $\|A\|$.*

Bemerkung 28 1. Die Konditionszahl hängt von der gewählten Norm ab. Wir bezeichnen mit

$$cond_p(A) \;:=\; \|A\|_p \cdot \|A^{-1}\|_p$$

die Werte für $p = 1, 2, \infty$. Die *spektrale Konditionszahl* lässt sich durch

$$cond_2(A) \;=\; \|A\|_2 \cdot \|A^{-1}\|_2 \;=\; \frac{\sigma_1(A)}{\sigma_n(A)} \tag{42}$$

mit Spektralnorm (39) und maximalem bzw. minimalem Singulärwert $\sigma_1(A)$ bzw. $\sigma_n(A)$ darstellen.

2. Wegen $\sigma_1(A) \geq \sigma_n(A)$ ist $cond(A) \geq 1$ für beliebige Matrizen A. Falls A singulär und damit $\sigma_n(A) = 0$ ist, setzt man $cond(A) = \infty$.

3. Mit der Normbeziehung (37) lässt sich die Konditionszahl durch

$$cond(A) \;=\; \max_{\|x\|=1} \|Ax\| \;/\; \min_{\|x\|=1} \|Ax\| \tag{43}$$

mit kompatibler Vektornorm beschreiben. Diese Darstellung gestattet eine einfache geometrische Interpretation der Konditionszahl. Die Matrix A vermittelt eine lineare Abbildung $y = Ax$ der Einheitssphäre $S(0;1) = \{x \in \mathbb{R}^n \mid \|x\| = 1\}$ in den Bildraum $\mathbb{R}^n$. Das Bild $W = \{y \in \mathbb{R}^n \mid \|y\| = 1,\, y = Ax\}$ kann man sich als deformierte (transformierte) Sphäre, z.B. als lang gestrecktes Ellipsoid W, vorstellen. Dann beschreibt $cond(A)$ das Verhältnis von größtem zu kleinstem Durchmesser dieser deformierten Fläche, woraus anschaulich $cond(A) \geq 1$ klar wird. □

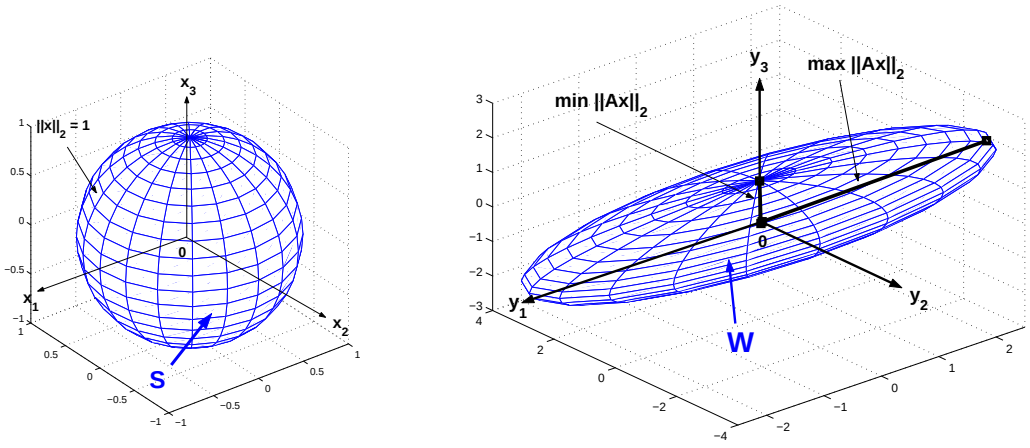


Abbildung 6: $cond_2(A)$

Wir wollen nun die gesuchte Fehlerschätzung herleiten. Ausgehend vom exakten Gleichungssystem

$$Ax \;=\; a \tag{44}$$

und dem gestörten System $A\tilde{x} = \tilde{a}$ erhält man durch Subtraktion ein System für den Fehler

$$A \cdot \Delta x \;=\; \Delta a \qquad \text{bzw.} \qquad \Delta x \;=\; A^{-1} \cdot \Delta a\,. \tag{45}$$

Übergang zu den Normen in (44) und (45) sowie Normabschätzung liefert

$$||\Delta x|| \;\leq\; ||A^{-1}|| \cdot ||\Delta a|| \qquad \text{und} \qquad ||a|| \;\leq\; ||A|| \cdot ||x||\,,$$

woraus wir durch Multiplikation

$$||\Delta x|| \cdot ||a|| \;\leq\; ||A|| \cdot ||A^{-1}|| \cdot ||x|| \cdot ||\Delta a|| \;=\; cond(A) \cdot ||x|| \cdot ||\Delta a||$$

erhalten. Nach Division ergibt sich hieraus die gesuchte Abschätzung für den relativen Lösungsfehler:

Satz 29 *Ist A regulär mit Fehler $\Delta A = 0$, so gilt mit beliebiger gegebener Norm $\|x\|$ und induzierter Matrixnorm $\|A\|$ die a-priori-Fehlerschätzung*

$$\frac{\|\Delta x\|}{\|x\|} \;\leq\; cond(A) \cdot \frac{\|\Delta a\|}{\|a\|}\,. \tag{46}$$

Die Konditionszahl ist offensichtlich ein Vergrößerungsfaktor für den relativen Fehler. Änderungen der rechten Gleichungssseiten in der Größe ε_a können im ungünstigsten Fall Änderungen der Lösung mit der Größenordnung $\varepsilon_x = cond(A)\varepsilon_a$ verursachen. Eine große Konditionszahl bedeutet ein *schlecht konditioniertes* Gleichungssystem.

Beispiel 30 Betrachten wir die symmetrische positiv definite Pascal-Matrix aus Beispiel 16 und 17. MATLAB berechnet bzw. schätzt folgende Konditionszahlen:

1. `cond(A)` oder `cond(A,2)` berechnet $cond_2(A)$ mittels der (aufwändigen) Singulärwertzerlegung.

2. `cond(A,1)` und `cond(A,inf)` berechnen $cond_1(A)$ und $cond_\infty(A)$ mittels der Matrixinversion `inv(A)` und sind damit effizienter als `cond(A,2)`.
3. `condest(A)` schätzt die Konditionszahl $cond_1(A)$ und benutzt u.a. eine LU-Zerlegung `lu(A)`.
4. `rcond(A)` schätzt $1/cond_1(A)$, ebenfalls mittels LU-Zerlegung `lu(A)`.

Für $n = 1, 2, \ldots, 30$ ist $cond_2(A)$ logarithmisch in Abb. 7 dargestellt. Bis zu $n = 18$ ist

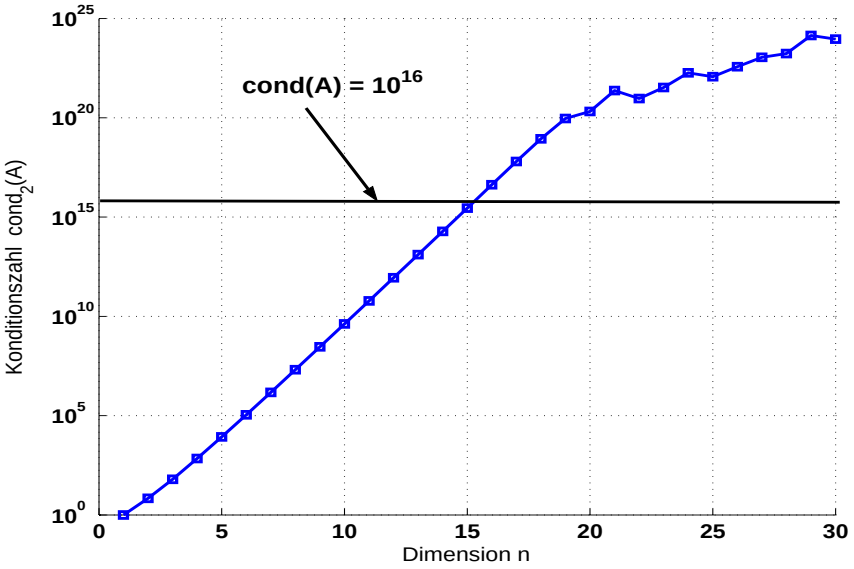


Abbildung 7: Relative Konditionszahl $cond_2(A)$ des Pascal-Systems

das exponentielle Anwachsen der Konditionszahl gut zu erkennen, während für größeres n die Konditionsberechnung selbst versagt. Als hpd-Matrizen erfüllen die Pascal-Matrizen die Beziehungen

$$cond_2(A) \;=\; \lambda_{max}/\lambda_{min} \qquad \text{und} \qquad cond_1(A) \;=\; cond_\infty(A)\,. \tag{47}$$

mit dem größten bzw. kleinsten Eigenwert λ_{max} bzw. λ_{min} der Matrix A. In Tabelle 1 werden die Konditionszahlen $cond_2(A)$ und $cond_\infty(A)$ dargestellt. Zudem wird die exakte Lösung x des Gleichungssystems $Ax = a$ aus Beispiel 17 mit einer (exakten) Lösung $\tilde{x}$ des Systems $A\tilde{x} = \tilde{a}$ verglichen, wobei $\tilde{a}_i = a_i + 100\varepsilon,\ i = 1(1)n$, mit der Maschinengenauigkeit ε gestört wurde. Die realen „Verstärkungsfaktoren“

$$\kappa(A) \;:=\; \frac{||\tilde{x} - x||_2}{||x||_2} \cdot \frac{||a||_2}{||\tilde{a} - a||_2}$$

sind zwar 3-4 Größenordnungen kleiner als die Konditionszahlen, sie wachsen allerdings auch bis auf 10^{15} an (vgl. Tabelle 1 und Abb. 8). Bei mehr als 20 Unbekannten ist eine Lösung dieses *schlecht konditionierten* Systems selbst in *double precision*-Genauigkeit nicht mehr möglich. □

Beispiel 31 Zum Vergleich erzeugen wir nun mit dem Kommando `A = rand(n)` reelle n-reihige Matrizen, deren Elemente gleichmäßig verteilte Zufallszahlen im Intervall $(0, 1)$ sind. Die relative Konditionszahlen $cond_1(A)$, $cond_2(A)$ und $cond_\infty(A)$ sind bis zu $N = 1000$ in Abb. 9 mit logarithmischer Skala dargestellt. Alle Konditionszahlen liegen unter 10^7, so dass die Lösung dieser *gut konditionierten* Systeme keine numerischen Probleme bereitet. □

n	$\kappa(A)$	$cond_2(A)$	$cond_\infty(A)$
1	1.0000e+000	1.0000e+000	1.0000e+000
5	3.6089e–002	8.5175e+003	1.5624e+004
10	3.7930e+005	4.1552e+009	8.1337e+009
15	4.8756e+011	2.8399e+015	5.7604e+015
20	1.1448e+015	2.0959e+020	1.7767e+021
25	6.8925e+014	1.1674e+022	9.8153e+022
30	8.2469e+014	9.1269e+023	1.3809e+025

Tabelle 1: Verstärkungsfaktoren und Konditionszahlen

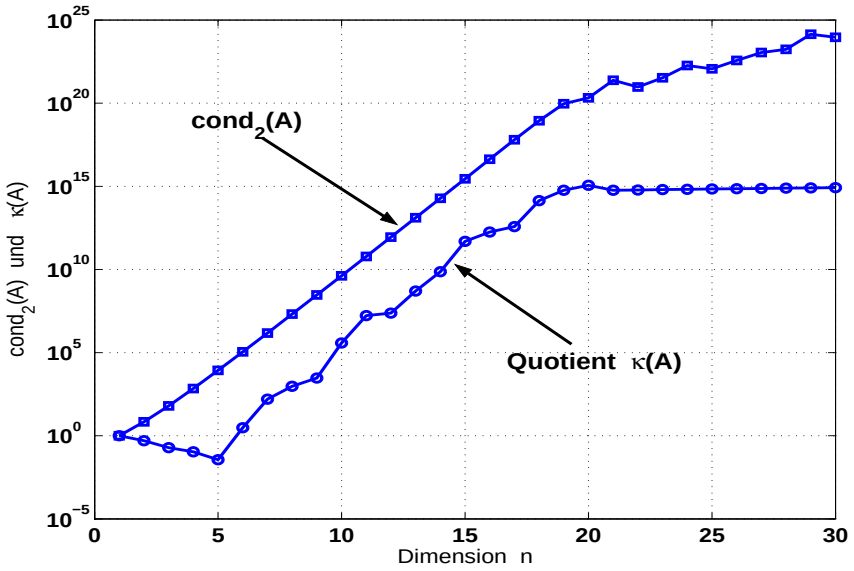


Abbildung 8: Verstärkungsfaktor und Konditionszahl des Pascal-Systems

Beispiel 32 Für $0 < \varepsilon \ll 1$ ist die Matrix $A = \begin{pmatrix} 1 & 1 \\ 1+\varepsilon & 1-\varepsilon \end{pmatrix}$ „fast singulär“, denn $\det(A) = -2\varepsilon \to 0$ für $\varepsilon \to 0$. Andererseits gilt dann auch

$$cond_\infty(A) \;=\; 1 + \frac{2}{\varepsilon} \;\to\; \infty\,.$$

Für die Nähe einer Matrix A zur Singularität stellt die Konditionszahl $cond(A)$ ein besseres Maß dar als deren Determinante. Denn wegen der allgemeinen Eigenschaft $cond(\lambda A) = cond(A)$, $\lambda \in \mathbb{K}$, besitzen die 100-reihige Einheitsmatrix $A = I_{100}$ und die Diagonalmatrix $B = 0.1 \cdot I_{100}$ bei induzierter Matrixnorm dieselbe Konditionszahl 1. Die Lösung $x_k = 10a_k$, $k = 1(1)100$ des linearen Gleichungssystems $Bx = a$ ist somit unproblematisch. Die Determinante $\det(B) = 10^{-100}$ dagegen suggeriert numerisch eine nicht vorhandene Singularität! □

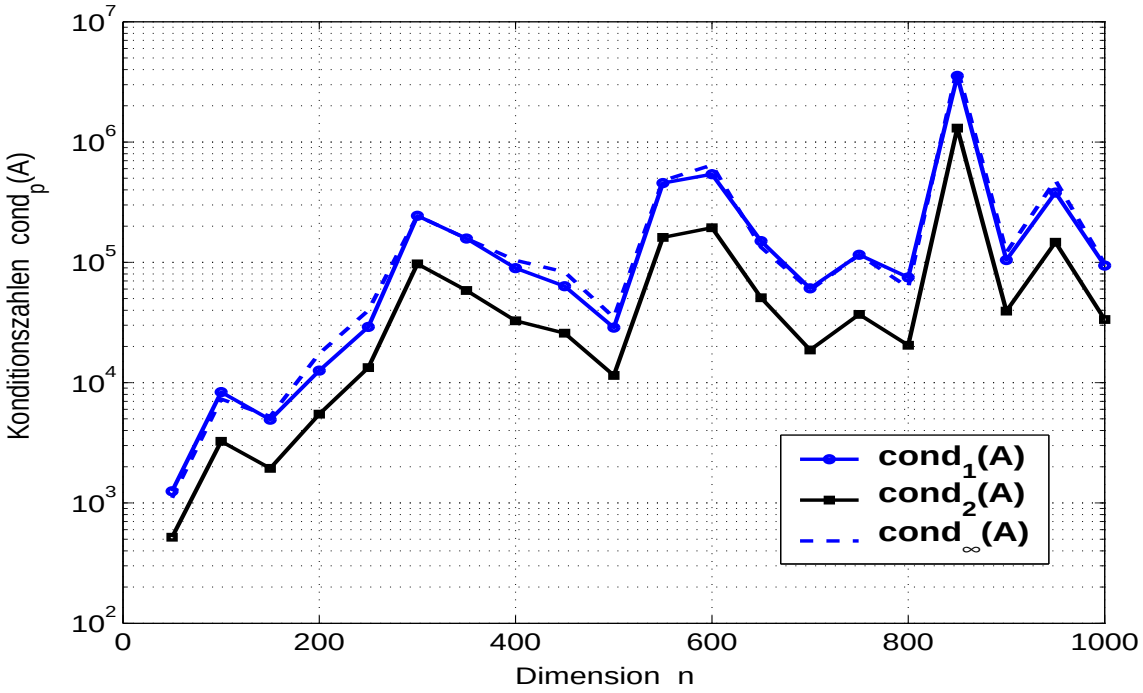


Abbildung 9: Relative Konditionszahlen von Zufallsmatrizen

Wenn die Matrixkoeffizienten A des linearen Gleichungssystems (44) ebenfalls fehlerbehaftet sind, also die Störungsmatrix ΔA nicht verschwindet, so lässt sich mit größerem Aufwand die a-priori-Fehlerschätzung (46) verallgemeinern. Den Beweis findet man z.B. in [16], S.39f.

Satz 33 *Sei A regulär mit Fehler ΔA, der die Bedingung*

$$\|\Delta A\| \;<\; 1/\|A^{-1}\| \tag{48}$$

erfüllen soll, d.h. der relative Koeffizientenfehler ist hinreichend klein. Dann gilt mit beliebiger gegebener Norm $\|x\|$ und induzierter Matrixnorm $\|A\|$ die verallgemeinerte a-priori-Fehlerschätzung

$$\frac{\|\Delta x\|}{\|x\|} \;\leq\; \frac{cond(A)}{1 - cond(A)\cdot \|\Delta A\| \,/\, \|A\|} \cdot \left[\frac{\|\Delta A\|}{\|A\|} + \frac{\|\Delta a\|}{\|a\|}\right] . \tag{49}$$

Beispiel 34 Wir betrachten das lineare System mit Lösung $x_1 = 0,\ x_2 = 1$

$$\begin{array}{rcl} x_1/99 \;+ x_2 &=& 1 \\ x_1/100 \;+ x_2 &=& 1\,, \end{array}$$

dessen Koeffizientenmatrix A die Norm $\|A\|_\infty = 100/99$ und die relative Konditionszahl $cond_\infty(A) = 20\,000$ besitzt. Stören wir nun $A_{12} = \varepsilon$ mit dem kleinen Wert $\varepsilon = 0.5 \cdot 10^{-4}$, so erhalten wir die fehlerbehaftete Koeffizientenmatrix und die Störung

$$\tilde{A} \;=\; \begin{pmatrix} 1/99 & 1+\varepsilon \\ 1/100 & 1 \end{pmatrix} \qquad \text{und} \qquad \Delta A \;=\; \begin{pmatrix} 0 & \varepsilon \\ 0 & 0 \end{pmatrix}$$

mit der Norm $\|\Delta A\|_\infty = \varepsilon$. Diese Störung erfüllt die Bedingung (48) des Satzes, der deshalb

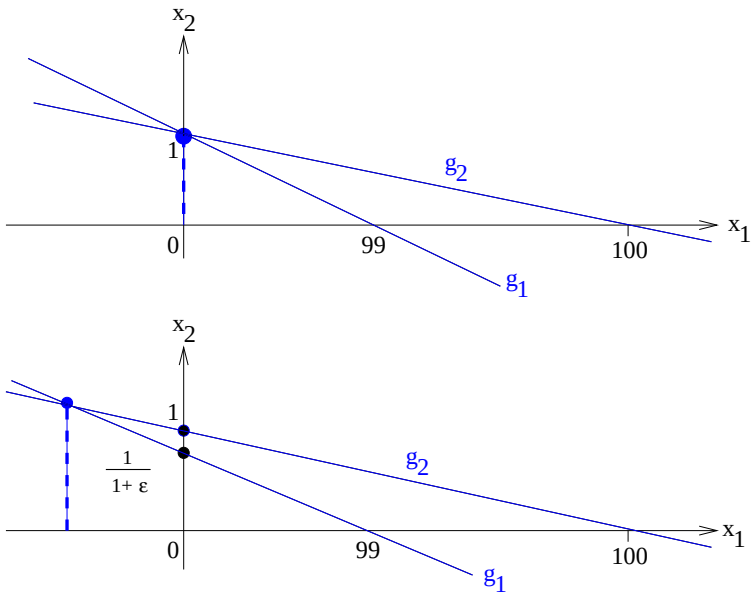


Abbildung 10: Auswirkung einer Störung ε

den Verstärkungsfaktor für den Fehler gemäß (49)

$$\kappa(A) \;:=\; \frac{cond_\infty(A)}{1 - cond_\infty(A) \cdot \|\Delta A\|_\infty \,/\, \|A\|_\infty} \;=\; 2\ 000\ 000$$

mit der ∞-Norm liefert. Auch wenn die rechten Seiten $a = (1,1)^T$ keinerlei Störung erfahren, weicht dennoch die erhaltene Lösung

$$\tilde{x}_1 \;=\; -0.4974624663, \quad \tilde{x}_2 \;=\; 1.004974625$$

des Systems $\tilde{A}x = a$ erheblich von der exakten Lösung $x = (0,1)^T$ ab. Eine Störung $\|\Delta A\| = 0.5 \cdot 10^{-4}$ der Matrixelemente bewirkt somit wegen $\|\Delta x\| = 0.497...$ eine 10 000-fache Abweichung der Lösung! Mit dem berechneten Verstärkungsfaktor $\kappa(A)$ überschätzt (49) den wahren Fehler allerdings um weitere 2 Größenordnungen.
Stellt man die 2 Gleichungen des Systems grafisch durch die Geraden g_1 und g_2 in Abb. 10 dar, so schneiden sie sich im exakten Lösungspunkt $(0,1)$ mit extrem spitzem Winkel. Die Einführung der Störung ε bewirkt eine Verschiebung des Schnittpunktes der Geraden g_1 mit der x_2-Achse um den kleinen Wert $\delta = \varepsilon/(1+\varepsilon)$ nach unten. Die Lösung in Abb. 10 verschiebt sich infolgedessen weit nach links in den Punkt $(-0.4974624663, 1.004974625)$. □

Allgemein entstehen schlecht konditionierte Systeme (50) in n Variablen oft dann, wenn sich die beschreibenden Hyperebenen in $\mathbb{R}^n$ unter extrem kleinem Winkel schneiden – ein Fall, der in der Praxis leider allzu oft auftritt.

1.5 Zusammenfassung

Direkte Verfahren liefern bis auf Rundungsfehler genau die exakte Lösung linearer Gleichungssysteme $Ax = a$ kleiner und mittlerer Dimension n in endlich vielen Rechenschritten.

1. Die *LU-Zerlegung (auch LR-Zerlegung genannt)* stellt die bekannteste Matrixfaktorisierung $A = LR$ dar. Sie zerlegt eine reguläre $n \times n$-Matrix A in das Produkt einer

unteren (L) und einer oberen (R) Dreiecksmatrix mit asymptotisch $\frac{2}{3}n^3$ Gleitpunkt-Operationen (flops).

2. Durch *Vorwärts- und Rückwärtssubstitution* kann jedes reguläre lineare Gleichungssystem $Ax = a$ nach erfolgter LU-Zerlegung von A mit lediglich $2n^2$ flops nach x aufgelöst werden.
3. Der *Gauß-Algorithmus* ist äquivalent zur LU-Zerlegung mit gleichzeitiger Transformation der rechten Seiten a (Vorwärtssubstitution) und anschließender Rückwärtssubstitution. Allerdings ist die kompaktere, Speicher-ökonomischere LU-Zerlegung dem Gauß-Algorithmus vorzuziehen.
4. Mittels *Pivotisierung* nimmt man geeignete Zeilenvertauschungen vor, um das Nullwerden von Diagonalelementen zu vermeiden. Mit der Spaltenmaximum-Strategie sichert man so betragskleine Elemente der L-Matrix und kann damit (theoretisch) jedes reguläre Gleichungssystem lösen. Das Wachstum der Einträge der R-Matrix hält sich bei praktischen Problemen in der Größenordung von $\sqrt{n}$.
5. Die *inverse Matrix* A^{-1} lässt sich bequem durch LU-Zerlegung ermitteln, allerdings mit dem 3-fachen Rechenaufwand von $2n^3$ flops. Sie sollte deshalb – soweit möglich – zugunsten der Lösung linearer Gleichungssysteme vermieden werden. Die Determinante $\det(A)$ ergibt sich mit Berücksichtigung der Zeilenvertauschungen aus den Pivotelementen von R .
6. Die *Cholesky-Zerlegung* ist ein Rundungsfehler-stabiles Verfahren zur Faktorisierung hermitescher (bzw. symmetrischer) positiv definiter $n \times n$-Matrizen. Lineare Gleichungssysteme mit derartigen hpd- (bzw. spd-) Koeffizientenmatrizen sollten mit dem Cholesky-Verfahren bei asymptotisch $\frac{1}{3}n^3$ flops – also mit halbiertem Rechenaufwand – gelöst werden.
7. Eine *Matrixnorm* $\|A\|$ ist eine reelle Kennzahl für die $n \times n$-Matrix A, welche die bekannten 3 Normeigenschaften besitzt. Die numerisch wichtigsten Normen sind die Zeilensummennorm, die Spaltensummennorm und die Frobeniusnorm. Sie finden oft bei Fehlerschätzungen Anwendung, in denen der Spektralradius $\varrho(A)$ von A nicht verfügbar ist.
8. Die *relative Konditionszahl* $\mathrm{cond}(A) \geq 1$ stellt einen Vergrößerungsfaktor für Eingangsfehler der Matrix A und der rechten Seiten a des linearen Gleichungssystems $Ax = a$ dar. Anwendung eines Verfahrens auf schlecht konditionierte Systeme mit cond $(A) > 10^{15}$ kann selbst bei kleinsten Eingangs- und Rundungsfehlern zu unbrauchbaren Resultaten führen. Hier hilft oft nur eine Rechnung in extrem hochgenauer Arithmetik.
9. *Weitere Matrixklassen*, wie diagonal-dominante Matrizen, Bandmatrizen, Toeplitz-Matrizen und unstrukturiert schwach besetzte Matrizen, werden zusammen mit weiteren direkten Verfahren in der Spezialliteratur [5, 9, 13, 20, 25, 28, 14, 32, 33] ausführlich behandelt.
10. Das *Numerik-System* MATLAB stellt hochentwickelte effiziente direkte Verfahren für reelle und komplexwertige lineare Systeme $Ax = a$ per Black-Box-Aufruf bereit. Diese sollten als Bausteine in eigene Anwendungen integriert werden.

2 Iterative Verfahren für großdimensionale lineare Gleichungssysteme

Wie im vorhergehenden Abschnitt betrachten wir normalbestimmte lineare Gleichungssysteme

$$Ax = a\,, \quad A \in \mathbb{C}^{n\times n},\ a, x \in \mathbb{C}^n \tag{50}$$

mit großdimensionaler Koeffizientenmatrix A und Vektoren a, x

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix},\ a = \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix},\ x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}.$$

Große Systeme treten zahlreich auf, z.B. bei der Näherungslösung gewöhnlicher und partieller Differenzialgleichungen in der Mechanik, aber auch unmittelbar bei der Modellierung von elektrischen Schaltkreisen. Die in Abschnitt 1 vorgestellten *direkten Verfahren* (Gauß-Algorithmus, LU-Zerlegung, Cholesky-Zerlegung) lösen die Aufgabe (50) bei gut konditionierten Systemen kleiner und mittlerer Dimension n in endlich vielen Schritten zuverlässig, effektiv und numerisch stabil. Dagegen übersteigt der Rechenzeitbedarf für großdimensionale Gleichungssysteme wegen der Zeitkomplexität $\mathcal{O}(n^3)$ der direkten Verfahren oft alle vertretbaren Grenzen. Eine Ausnahme bilden gegebenenfalls schwach besetzte Matrizen, z.B. die bei Diskretisierungen auftretenden Bandmatrizen und Block-Bandmatrizen (vgl. [13]).

Im allgemeinen Fall vollbesetzter (oder sehr großer schwachbesetzter) Koeffizientenmatrizen sind deshalb *iterative Verfahren* die bessere Alternative, da der Aufwand pro Iterationsschritt mit $\mathcal{O}(n^2)$ für eine Matrix-Vektor-Multiplikation beträchtlich geringer ist und deshalb Verfahren mit $k \ll n$ Iterationen durchweg den direkten Lösern überlegen sind. Iterative Verfahren werden gegenwärtig in einer Vielzahl von Varianten eingesetzt. Nachfolgend werden die für die Praxis bedeutsamen *Aufspaltungsverfahren (auch: Splitting-Verfahren)* und ausgewählte *Projektionsverfahren* vorgestellt.

2.1 Splitting-Verfahren

Lineare Iterationsverfahren Um den Banachschen Fixpunktsatz (vgl. [15], Kapitel 14) auf den Spezialfall linearer Gleichungssysteme anwenden zu können, ist das Ausgangssystem (50) in Fixpunktform (iterierfähige Form) zu transformieren. Dies kann hier systematisch durchgeführt werden, indem man die Gleichungen nach ihren Hauptdiagonalelementen auflöst:

$$\begin{aligned}
x_1 &= \frac{1}{a_{11}}\,(a_1 \qquad\quad - a_{12}x_2 - \cdots - a_{1,n-1}x_{n-1} - a_{1n}x_n) \\
x_2 &= \frac{1}{a_{22}}\,(a_2 - a_{21}x_1 \qquad\quad - \cdots - a_{2,n-1}x_{n-1} - a_{2n}x_n) \\
&\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\
x_n &= \frac{1}{a_{nn}}\,(a_n - a_{n1}x_1 - a_{n2}x_2 - \cdots - a_{n,n-1}x_{n-1} \qquad\quad).
\end{aligned} \tag{51}$$

Falls gewisse a_{ii} verschwinden, sind geeignete Zeilenvertauschungen vorzunehmen. Ausgehend von einer Startnäherung $x_0 = (x_{01}, x_{02}, \ldots, x_{0n})^T$ erhält man durch Einsetzen in die rechte Seite eine neue Näherungslösung $x_1 = (x_{11}, x_{12}, \ldots, x_{1n})^T$ und bei k-maliger Wiederholung die Iterationsvorschrift

$$x_{k+1,i} = \frac{1}{a_{ii}} \left(a_i - \sum_{\substack{j=1 \\ j \neq i}}^{n} a_{ij} x_{kj} \right) , \quad i = 1(1)n \quad \text{für} \quad k = 0, 1, 2, ...$$

Diese als *Jacobi-Verfahren* oder *Gesamtschrittverfahren* bekannte Fixpunktiteration kann man leicht verallgemeinern. Wir setzen dazu voraus, dass $\det(A) \neq 0$ ist und damit das lineare Gleichungssystem (50) eine eindeutige Lösung x_* besitzt. Dann lässt sich die Koeffizientenmatrix auf beliebig viele Arten in der Form

$$A = N - P \qquad \text{mit} \qquad \det(N) \neq 0 \tag{52}$$

aufspalten (engl. *splitting method*), wobei N und P ebenfalls (n, n)-Matrizen sind. Das System kann nun in

$$Nx = Px + a \qquad \text{bzw. in} \qquad x = N^{-1}Px + N^{-1}a$$

umgeformt werden. Mit den Bezeichnungen $B := N^{-1}P$ und $b = N^{-1}a$ erhalten wir daraus das äquivalente System in *Fixpunktform (iterierfähiger Form)*

$$x = Bx + b\,. \tag{53}$$

Beginnend mit einem Startvektor $x_0 \in \mathbb{C}^n$, gewinnt man so eine Folge (x_k) von Vektoren durch

$$\boxed{x_{k+1} \;=\; Bx_k + b\,, \quad k = 1, 2, 3, ...} \tag{54}$$

In der affinen Abbildung (der Iterationsfunktion) $g : \mathbb{C}^n \to \mathbb{C}^n$ mit

$$g(x) = Bx + b$$

treten die Komponenten von x und b linear auf, so dass hier ein *stationäres lineares Einschrittverfahren* vorliegt.

Definition 35 (Lineares Iterationsverfahren)

(*i*) *Ein Iterationsverfahren* $x_{k+1} = g(x_k)\,,\ k = 0, 1, \ldots$ *heißt lineares Einschrittverfahren, wenn* g *die Form* $g(x) = Bx + b$ *besitzt.*

(*ii*) *Die Matrix* $B = N^{-1}P$ *heisst Iterationsmatrix des Iterationsverfahrens.*

(*iii*) *Das Iterationsverfahren (54) ist stationär, wenn* B *und* b *nicht vom Iterationsindex* k *abhängen.*

Wegen der Darstellung $g(x) = Bx + b$ ist die Abbildung g stetig differenzierbar mit der konstanten Jacobi-Matrix $G(x) := g'(x) = B$. Damit vereinfachen sich die Aussagen des Banachschen Fixpunktsatzes und liefern folgenden

Satz 36 *Mit einer beliebigen gegebenen Matrixnorm sei* $||B|| < 1$ *erfüllt. Dann gelten folgende Behauptungen:*

(*i*) *Gleichung (53) besitzt eine eindeutige Lösung* x_**, den einzigen Fixpunkt von* g*. Dies ist die eindeutige Lösung des linearen Gleichungssystems* $Ax = a$*.*

(*ii*) *Die Iterationsvektoren* x_k *des Verfahrens (54) konvergieren bei beliebig gewähltem Startvektor* $x_0 \in \mathbb{C}^n$ *stets gegen die Lösung* x_* *.*

(*iii*) *Für den k-ten Iterationsvektor* x_k *gilt mit* $\lambda := ||B||$ *die a-priori-Fehlerschätzung*

$$\|x_k - x_*\| \leq \frac{\lambda^n}{1-\lambda}\|x_1 - x_0\| \tag{55}$$

sowie die a-posteriori-Fehlerschätzung

$$\|x_k - x_*\| \leq \frac{\lambda}{1-\lambda}\|x_k - x_{k-1}\| \; . \tag{56}$$

Beweis: Mit der Kontraktionskonstanten $\lambda := ||G(x)|| = ||B|| < 1$ ist der Banachsche Fixpunktsatz für die Gleichung $x = g(x)$ global auf ganz $\mathbb{C}^n$ erfüllt. An die Startlösung x_0 muss keine einschränkende Bedingung gestellt werden, so dass der Satz nun in $\mathbb{C}^n$ gültig ist, womit auch die Fehlerschätzungen folgen. □

Der zugehörige Algorithmus 37 für das lineare Einschrittverfahren (54) setzt als Input die Iterationsmatrix B, den Vektor b, einen Startwert x und die (absolute und relative) Toleranz $tolabs, tolrel$ voraus. Für die Konvergenz sind offenbar allein die Eigenschaften der Iterationsmatrix B, nicht aber die des Vektors b ausschlaggebend. Es genügt, wenn für irgendeine Matrixnorm der Nachweis $\|B\| < 1$ erfolgt. Das kann auch für eine kompatible Norm, z.B. die Frobeniusnorm $\|B\|_F$ gelten, denn wegen $\|B\|_2 \leq \|B\|_F < 1$ ist dann die Voraussetzung mit der Spektralnorm $\|B\|_2$ erfüllt und die Konvergenz in der Euklidischen Vektornorm

Algorithmus 37 (Lineares Einschrittverfahren)

Function $[y] =$ linear$(B, b, x, tolabs, tolrel)$

1. Berechne eine Norm $\lambda = ||B||$. Falls $\lambda > 1$, so Stop.
2. Berechne $y = Bx + b$ und die Toleranz $tol = tolrel \cdot ||y|| + tolabs$
3. Do while $\frac{\lambda}{1-\lambda}||y - x|| > tol$
   1. Überspeichere $x = y$
   2. Berechne $y = Bx + b$
   3. Aktualisiere $tol = tolrel \cdot ||y|| + tolabs$
4. Return y

garantiert. Mit den speziellen Matrixnormen $||B||_\infty$ oder $||B||_1$ lauten die Konvergenzbedingungen

$$\|B\|_\infty \;=\; \max_{i=1(1)n} \;\sum_{j=1}^{n} |b_{ij}| \;<\; 1 \tag{57}$$

bzw.

$$\|B\|_1 \;=\; \max_{j=1(1)n} \;\sum_{i=1}^{n} |b_{ij}| \;<\; 1\,. \tag{58}$$

Diese Normbedingungen sind hinreichend, nicht jedoch notwendig für die Konvergenz gegen eine Lösung. Ein notwendiges und hinreichendes Konvergenzkriterium erhält man mit Hilfe des Spektralradius $\varrho(B)$ der Iterationsmatrix B. Den Beweis findet man z.B. in [12], S.59.

Satz 38 *Die Gleichung $x = Bx + b$ besitzt eine eindeutige Lösung $x_* \in \mathbb{C}^n$, und die Iterationsvektoren x_k der Iteration (54) konvergieren bei beliebig gewähltem Startvektor $x_0 \in \mathbb{C}^n$ gegen die Lösung x_* genau dann, wenn $\varrho(B) < 1$ ist.*

Iteration in Gesamtschritten (Jacobi-Verfahren) Die nachfolgenden Iterationsverfahren sind Spezialfälle des allgemeinen Verfahrens

$$x_k = Bx_{k-1} + b\,, \quad B = N^{-1}P, \quad b = N^{-1}a$$

und unterscheiden sich lediglich in der Wahl von N und P. Das in (51) beschriebene Verfahren einer Auflösung nach den Hauptdiagonalelementen von A ergibt die Matrixdarstellung

$$\underbrace{\begin{pmatrix} a_{11} & 0 & \cdots & 0 \\ 0 & a_{22} & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & a_{nn} \end{pmatrix}}_{N} \begin{pmatrix} x_1 \\ x_2 \\ \cdot \\ x_n \end{pmatrix} = \underbrace{\begin{pmatrix} 0 & -a_{12} & \cdots & -a_{1n} \\ -a_{21} & 0 & \cdots & -a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ -a_{n1} & -a_{n2} & \cdots & 0 \end{pmatrix}}_{P} \begin{pmatrix} x_1 \\ x_2 \\ \cdot \\ x_n \end{pmatrix} + \underbrace{\begin{pmatrix} a_1 \\ a_2 \\ \cdot \\ a_n \end{pmatrix}}_{a}$$

A wird also in die Diagonalmatrix N und die Restmatrix P zerlegt. Das *Gesamtschrittverfahren (Jacobi-Verfahren, J-Verfahren)* lautet dann

$$\boxed{x_{k,i} \;=\; \frac{1}{a_{ii}} \left(a_i - \sum_{\substack{j=1 \\ j\neq i}}^{n} a_{ij} x_{k-1,j} \right), \quad i = 1(1)n\,.} \tag{59}$$

Es erhielt seinen Namen dank der Berechnung des kompletten Iterationsvektors x_k im k-ten „Gesamtschritt“ aus dem vorhergehenden Vektor x_{k-1}. Erst danach ist eine Überspeicherung dieses Vektors durch den neuen Iterationsvektor möglich. Da die Iterationsmatrix $B = N^{-1}P$ nun die Elemente

$$b_{ij} = \begin{cases} -a_{ij}/a_{ii} & \text{für} \quad i \neq j \\ 0 & \text{für} \quad i = j \end{cases}$$

besitzt, liefert die Bedingung $||B||_p < 1$ mit den Matrixnormen für $p = \infty, 1, F$ die Darstellungen

$$\sum_{\substack{j=1\\ j\neq j}}^{n} \left|\frac{a_{ij}}{a_{ii}}\right| < 1 \quad \text{für alle} \quad i = 1(1)n \qquad (\textit{starkes Zeilensummenkriterium}), \tag{60}$$

$$\sum_{\substack{i=1\\ i\neq j}}^{n} \left|\frac{a_{ij}}{a_{ii}}\right| < 1 \quad \text{für alle} \quad j = 1(1)n \qquad (\textit{starkes Zeilensummenkriterium}), \tag{61}$$

$$\sum_{i=1}^{n} \sum_{\substack{j=1\\ j\neq i}}^{n} \left|\frac{a_{ij}}{a_{ii}}\right|^2 < 1 \qquad (\textit{Quadratsummenkriterium}). \tag{62}$$

Beispiel 39 Für das lineare reelle System

$$\begin{aligned} 12x_1 + 2x_2 + 3x_3 &= 18 \\ -x_1 + 8x_2 + 2x_3 &= -32 \\ x_1 + 3x_2 + 12x_3 &= 6 \end{aligned}$$

lautet das Gesamtschrittverfahren

$$\begin{aligned} x_{k+1,1} &= 1.5 - 0.1667\, x_{k,2} - 0.25\, x_{k,3} \\ x_{k+1,2} &= -4.0 + 0.125\, x_{k,1} - 0.25\, x_{k,3} \\ x_{k+1,3} &= 0.5 - 0.0833\, x_{k,1} + 0.25\, x_{k,2}\,. \end{aligned}$$

Bezeichnet man $b_i = a_i/a_{ii}$, $b_{ij} = -a_{ij}/a_{ii}$, so erhält man für die Koeffizienten b_{ij} das folgende Schema mit dem Startvektor $x_0 = 0$:

b_i	b_{i1}	b_{i2}	b_{i3}
1.5	0	-0.1667	-0.2500
-4.0	0.125	0	-0.2500
0.5	-0.0833	0.2500	0
k	$x_{k,1}$	$x_{k,2}$	$x_{k,3}$
0	0	0	0
1	1.5000	-4.0000	0.5000
2	2.0418	-3.9375	-0.6250
3	2.3126	-3.5885	-0.6545
4	2.2618	-3.5473	-0.5898
5	2.2388	-3.5698	-0.5752
6	2.2389	-3.5764	-0.5789
7	2.2409	-3.5754	-0.5806
8	2.2411	-3.5747	-0.5805
9	2.2410	-3.5747	-0.5804
10	2.2410	-3.5748	-0.5804

Näherungslösungen sind damit $x_1 = 2.2410$, $x_2 = -3.5748$, $x_3 = -0.5804$. Für die Iterationsmatrix

$$B = \begin{pmatrix} 0 & -0.1667 & -0.25 \\ 0.125 & 0 & -0.25 \\ -0.0833 & 0.25 & 0 \end{pmatrix}$$

ist wegen

$$\begin{aligned} \|B\|_\infty &= \max_i \sum_j |b_{ij}| = 0.4167 < 1 \qquad \text{und} \\ \|B\|_1 &= \max_j \sum_i |b_{ij}| = 0.5 < 1 \end{aligned}$$

die Konvergenz bei beliebiger Startnäherung garantiert. In der Maximumnorm $\|x\|_\infty$ ergibt sich mit den Konstanten $\lambda = \|B\|_\infty = 0.4167$ und $\|x_1 - x_0\| = 4.0$ die a-priori-Abschätzung

$$\|x_k - x_*\|_\infty \;\le\; \frac{0.4167^k}{1 - 0.4167} \cdot 4 \;=\; \alpha_k \,.$$

Durch Vergleich dieser Schranken α_k mit den tatsächlichen Fehlergrößen

k	α_k	$\|x_k - x_*\|_\infty$
2	1.191	0.3627
4	0.2068	0.0275
6	0.0359	0.0021
8	0.0062	0.0001

erkennt man die starke Überschätzung des Fehlers durch die a-priori-Schätzung. □

Das Gesamtschrittverfahren kann in Matrixschreibweise angegeben werden, wenn man die additive Aufspaltung von A mittels $A = D + L + R$ in die Diagonalmatrix
$D = \mathrm{diag}(a_{11}, a_{22}, ..., a_{nn})$ und die Dreiecksmatrizen

$$L = \begin{pmatrix} 0 & \cdots & & \\ a_{21} & 0 & & \\ \vdots & & & \\ a_{n1} & a_{n2} & \cdots & 0 \end{pmatrix}, \qquad R = \begin{pmatrix} 0 & a_{12} & \cdots & a_{1n} \\ 0 & 0 & \cdots & a_{2n} \\ \vdots & \vdots & & \\ 0 & 0 & \cdots & 0 \end{pmatrix}$$

vornimmt. Mit $N = D$ und $P = -(L + R)$ lautet das Verfahren dann

$$\boxed{x^{k+1} = D^{-1}[a - (L + R)x^k], \quad k = 0, 1, 2, \ldots .} \tag{63}$$

mit der Iterationsmatrix $B = -D^{-1}(L + R)$.

Iteration in Einzelschritten (Gauß-Seidel-Verfahren) Wendet man zur Bestimmung des skalaren Iterationswertes $x_{k,i}$ jeweils die zuletzt bestimmten Werte aller anderen Komponenten x_j an, so ergibt sich folgende Verfahrensvariante:

$$\begin{aligned} x_{k,1} &= \frac{1}{a_{11}}(a_1 - a_{12}x_{k-1,2} - a_{13}x_{k-1,3} - \ldots - a_{1n}x_{k-1,n}) \\ x_{k,2} &= \frac{1}{a_{22}}(a_2 - a_{21}x_{k,1} - a_{23}x_{k-1,3} - \ldots - a_{1n}x_{k-1,n}) \\ x_{k,3} &= \frac{1}{a_{33}}(a_3 - a_{31}x_{k,1} - a_{32}x_{k,2} - \ldots - a_{1n}x_{k-1,n}) \\ &\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ x_{k,n} &= \frac{1}{a_{nn}}(a_n - a_{n1}x_{k,1} - a_{n2}x_{k,2} - \ldots - a_{n,n-1}x_{k,n-1}) \,. \end{aligned} \tag{64}$$

In Kurzform lautet dieses *Gauß-Seidel-Verfahren (Einzelschrittverfahren)*

$$\boxed{x_{k,i} \;=\; \frac{1}{a_{ii}}\left(a_i - \sum_{j=1}^{i-1} a_{ij}x_{k,j} - \sum_{j=i+1}^{n} a_{ij}x_{k-1,j}\right), \quad i = 1(1)n} \tag{65}$$

für $k = 1, 2, 3, \ldots$. Da man nach Bestimmung einer Komponente $x_{k,i}$ des Näherungsvektors x_k den im Speicher stehenden Vektor x aktualisiert, wird x_k in jedem „Einzelschritt“ verändert, woher auch der Name *Einzelschrittverfahren* kommt. Aus (65) folgt übrigens

$$a_{ii}x_{k,i} + \sum_{j=1}^{i-1} a_{ij}x_{k,j} + \sum_{j=i+1}^{n} a_{ij}x_{k-1,j} \;=\; a_i$$

und damit die Darstellung in Matrixform

$$\begin{pmatrix} a_{11} & & & 0 \\ & a_{22} & & \\ & & \cdot & \\ 0 & & & a_{nn} \end{pmatrix}\begin{pmatrix} x_{k1} \\ x_{k2} \\ \cdots \\ x_{kn} \end{pmatrix} + \begin{pmatrix} 0 & 0 & \cdots & 0 \\ +a_{21} & 0 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ +a_{n1} & +a_{n2} & \cdots & 0 \end{pmatrix}\begin{pmatrix} x_{k1} \\ x_{k2} \\ \cdots \\ x_{kn} \end{pmatrix} -$$

$$- \begin{pmatrix} 0 & -a_{12} & \cdots & -a_{1n} \\ 0 & 0 & \cdots & -a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 0 \end{pmatrix}\begin{pmatrix} x_{k-1,1} \\ x_{k-1,2} \\ \cdots \\ x_{k-1,n} \end{pmatrix} = \begin{pmatrix} a_1 \\ a_2 \\ \cdots \\ a_n \end{pmatrix}.$$

Die Matrix A wird also in folgende Matrizen zerlegt:

$$A = \underbrace{\begin{pmatrix} a_{11} & & & & \\ a_{21} & a_{22} & & 0 & \\ a_{31} & a_{32} & a_{33} & & \\ \cdots & \cdots & \cdots & \cdots & \\ a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nn} \end{pmatrix}}_{N} - \underbrace{\begin{pmatrix} 0 & -a_{12} & -a_{13} & \cdots & -a_{1n} \\ 0 & 0 & -a_{23} & \cdots & -a_{2n} \\ 0 & 0 & 0 & \cdots & -a_{3n} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & 0 \end{pmatrix}}_{P}$$

$$A = \qquad\qquad N \qquad\qquad - \qquad\qquad P$$

Offenbar ist $\det(N) \neq 0$, falls $a_{ii} \neq 0$, $i = 1(1)n$ ist. Dass die Konvergenzkriterien (60) und (61) des Gesamtschrittverfahrens auch hinreichend für die Konvergenz des Gauß-Seidel-Verfahrens sind, zeigt folgender

Satz 40 *Starkes Zeilensummenkriterium (60) und starkes Spaltensummenkriterium (61) sind hinreichend für die Konvergenz des Gauß-Seidel-Verfahrens.*

Beweis: Wir beweisen die Behauptung für das starke Zeilensummenkriterium. Mit $b_{ij} := a_{ij}/a_{ii}$, $b_i := a_i/a_{ii}$ lautet das Verfahren

$$x_{k,i} = b_i - \sum_{j=1}^{i-1} b_{ij}x_{k,j} - \sum_{j=i+1}^{n} b_{ij}x_{k-1,j}\,.$$

Die Lösung x_* erfüllt die Fixpunktgleichung

$$x_{*,i} = b_i - \sum_{j=1}^{i-1} b_{ij} x_{*,j} - \sum_{j=i+1}^{n} b_{ij} x_{*,j} \, .$$

Subtraktion beider Beziehungen liefert für die Fehlergrößen $e_{k,i} := x_{k,i} - x_{*,i}$ die Abschätzungen

$$\begin{aligned} e_{k,i} &= -\sum_{j=1}^{i-1} b_{ij} e_{k,j} - \sum_{j=i+1}^{n} b_{ij} e_{k-1,j} \\ |e_{k,i}| &\le \sum_{j=1}^{i-1} |b_{ij}||e_{k,j}| + \sum_{j=i+1}^{n} |b_{ij}||e_{k-1,j}| \\ &\le \sum_{j=1}^{i-1} |b_{ij}| \cdot \|e_k\|_\infty + \sum_{j=i+1}^{n} |b_{ij}| \cdot \|e_{k-1}\|_\infty \\ |e_{k,i}| &\le p_i \cdot \|e_k\|_\infty + q_i \cdot \|e_{k-1}\|_\infty, \quad i = 1(1)n \end{aligned}$$

mit $p_i = \sum_{j=1}^{i-1} |b_{ij}|, \quad q_i = \sum_{j=i+1}^{n} |b_{ij}| \, .$

Sei $s = s(k)$ derjenige Indexwert für i, für den $|e_{k,s}| = \max |e_{k,i}| = \|e_k\|_\infty$ angenommen wird. Dann gilt offenbar für $i = s$ wegen Voraussetzung $p_s < 1$

$$\|e_k\|_\infty \le p_s \cdot \|e_k\|_\infty + q_s \|e_{k-1}\|_\infty \qquad \text{bzw.} \qquad \|e_k\|_\infty \le \frac{q_s}{1-p_s} \|e_{k-1}\|_\infty \, .$$

Es bleibt nun noch zu zeigen, dass mit der Iterationsmatrix B des *Gesamtschrittverfahrens*

$$\lambda := \max_s \frac{q_s}{1-ps} \le \|B\|_\infty = \max_i \sum_{\substack{j=1 \\ j \ne i}}^{n} |b_{ij}| < 1$$

gilt. Da aber nach Voraussetzung $p_i + q_i = \sum_{\substack{j=1 \\ j \ne i}}^{n} |b_{ij}| \le \|B\|_\infty < 1$ ist, folgt $q_i \le \|B\|_\infty - p_i$ und hieraus

$$\frac{q_i}{1-p_i} \le \frac{\|B\|_\infty - p_i}{1-p_i} \le \frac{\|B\|_\infty - p_i \|B\|_\infty}{1-p_i} = \|B\|_\infty .$$

Deshalb ist mit $\lambda := \max_i \frac{q_i}{1-p_i}$ die Ungleichung $\lambda \le \|B\|_\infty < 1$ erfüllt. Der Beweis für das starke Spaltensummenkriterium verläuft ähnlich. □

Folgerung 41 *Wegen* $\lambda := \max_s (q_s/(1-p_s)) \le \|B\|_\infty$ *konvergiert das Einzelschrittverfahren bei Erfülltsein des starken Zeilensummenkriteriums nicht langsamer als das Gesamtschrittverfahren.*

Beispiel 42 Wir betrachten Beispiel 39. Das Einzelschrittverfahren lautet

$$\begin{aligned} x_{k+1,1} &= 1.5 && - 0.1667\, x_{k,2} && - 0.2500\, x_{k,3} \\ x_{k+1,2} &= -4.0 + 0.125\, x_{k+1,1} && && - 0.2500\, x_{k,3} \\ x_{k+1,3} &= 0.5 - 0.0833\, x_{k+1,1} && + 0.2500\, x_{k+1,2} && \end{aligned}$$

Mit dem Schema

b_i	b_{i1}	b_{i2}	b_{i3}
1.5	*	-0.1667	-0.2500
-4.0	0.125	*	-0.2500
0.5	-0.0833	0.2500	*
k	$x_{k,1}$	$x_{k,2}$	$x_{k,3}$
0	0	0	0
1	1.5	-5.8125	0.5781
2	2.27996	-3.5705	-0.5826
3	2.2407	-3.5743	-0.5803
4	2.2408	-3.5748	-0.5804
5	2.2409	-3.5748	-0.5804

erhält man nach 5 Iterationen die Näherungen

$$x_1 = 2.2409, \quad x_2 = -3.5748, \quad x_3 = -0.5804\,.$$

Das Zeilensummenkriterium liefert $\|B\|_\infty = 0.4167$. Am Schema erkennt man, dass der Konvergenzfaktor $\lambda = \max\limits_s \frac{q_s}{1-p_s} = 0.4167$ nicht kleiner als beim Gesamtschrittverfahren ist. Dennoch konvergiert das Gauß-Seidel-Verfahren mit 5 Schritten wesentlich rascher als das Jacobi-Verfahren, das 10 Schritte benötigt. □

Zur Gewinnung der Matrixschreibweise des Einzelschrittverfahrens zerlegt man die Koeffizientenmatrix additiv $A = D + L + R$ mit den bereits eingeführten Matrizen und erhält

$$\begin{aligned} Dx + Lx + Rx &= a \\ (D+L)x &= a - Rx \\ x &= (D+L)^{-1}[a - Rx]\,, \end{aligned}$$

womit $N = D + L$ und $P = -R$ gilt und damit die Iterationsmatrix $B = N^{-1}P = -(D+L)^{-1}R$ lautet. Aufgelöst nach x^{k+1} ergibt sich die erste Darstellungsform des Gauß-Seidel-Verfahrens

$$\boxed{x_{k+1} = (D+L)^{-1}[a - Rx_k], \quad k = 0, 1, 2, \ldots.} \tag{66}$$

Wegen der aufwändigen Matrixinversion $(D+L)^{-1}$ ist diese Darstellung keinesfalls für praktische Rechnungen zu empfehlen; hier sollte die zweite Form (65) in Matrixform

$$x_{k+1} = D^{-1}[a - Lx_{k+1} - Rx_k], \quad k = 0, 1, 2, \ldots$$

benutzt werden.

2.2 Systeme mit spezieller Struktur und Relaxation

Diagonaldominanz Die Fixpunktiteration für lineare Gleichungssysteme erfordert eine solche iterierfähige Form, die ein konvergentes Verfahren liefert. Für zahlreiche Anwendungsprobleme ist eine geeignete Umstellung tatsächlich leicht möglich, insbesondere für strikt diagonal-dominante Matrizen.

Definition 43 (Strikte Diagonaldominanz) *$A \in \mathbb{C}^{n\times n}$ ist strikt diagonal-dominant, falls*

$$\sum_{\substack{j=1\\ j\neq i}}^{n} |a_{ij}| \;<\; |a_{ii}|\,, \quad i = 1(1)n \qquad \textit{gilt.}$$

Offenbar sind die Diagonalelemente a_{ii} derartiger Koeffizientenmatrizen stets ungleich Null, so dass Gesamt- und Einzelschrittverfahren ohne Zeilenvertauschung anwendbar sind. Darüber hinaus ist offensichtlich das starke Zeilensummenkriterium

$$\sum_{\substack{j=1\\ j\neq j}}^{n} \left|\frac{a_{ij}}{a_{ii}}\right| < 1 \qquad \text{für alle } i = 1(1)n$$

erfüllt, womit nach Satz 36 und Satz 40 beide Iterationsverfahren gegen die eindeutige Lösung x^* konvergieren. Damit hat man folgenden Satz bewiesen:

Satz 44 *Ist die Koeffizientenmatrix A strikt diagonal-dominant, so ist A regulär und das lineare Gleichungssystem eindeutig lösbar. Gesamt- und Einzelschrittverfahren sind konvergent, wobei das Einzelschrittverfahren nicht langsamer konvergiert.*

Das konstruierte System in Beispiel 39 ist offenbar strikt diagonal-dominant. Diese günstige Eigenschaft tritt häufig bei Problemklassen linearer Differenzialgleichungen auf.

Beispiel 45 Wir betrachten lineare Randwertprobleme 2. Ordnung der Form

$$-\frac{d^2x}{dt^2} + p(t)x = q(t)\,, \quad x(a) = x_a,\; x(b) = x_b \tag{67}$$

auf dem Intervall $I = [a, b]$. Die Funktionen p, q seien stetig, und es existiere eine Konstante $Q > 0$, so dass $p(t) \geq Q > 0$ auf I gilt. Dann besitzt das Problem nach [16] eine eindeutige Lösung $x \in C^2(I)$. Man kann diese Lösung $x(t)$ an den Gitterpunkten $t_i = a+ih,\; i = 0(1)n$, mit der Schrittweite $h = (b-a)/n$ durch Näherungen x_i approximieren, indem man das gegebene Problem durch die endlichdimensionale Aufgabe am Gitterpunkt t_i

$$-\frac{1}{h^2}(x_{i-1} - 2x_i + x_{i+1}) + p(t_i)x_i \;=\; q(t_i)$$

ersetzt. Mit den Abkürzungen $a_i := 2 + h^2p(t_i)$ und $b_i := h^2q(t_i)$ ergibt sich nach Zusammenfassung ein lineares Gleichungssystem

$$-x_{i-1} + a_ix_i - x_{i+1} \;=\; b_i\,, \quad i = 1(1)n-1\,. \tag{68}$$

Die beiden Lösungskomponenten $x_0 = x_a$ und $x_n = x_b$ sind durch die Randbedingungen x_a und x_b vorgegeben und können in die erste und letzte Gleichung eingesetzt werden. Um eine genaue Approximation der Lösung zu erhalten, ist n hinreichend groß zu wählen (z.B. $n = 500$). Die Koeffizientenmatrix A hat tridiagonale Form und ist wegen der Abschätzung $|a_i| = 2 + h^2p(t_i) \geq 2 + h^2Q > 2$ strikt diagonal-dominant. Damit ist System (68) bei beliebiger fester Dimension n eindeutig lösbar und beide Iterationsverfahren konvergieren gegen diese Lösung. Allerdings wird die Konvergenz wegen

$$||B||_\infty = \max_i \left|\frac{2}{a_i}\right| = \max_i \frac{2}{2 + h^2p(t_i)} \;\to\; 1 \quad \text{für} \quad h \to 0$$

bei höherer Approximationsgenauigkeit n zunehmend langsamer verlaufen. □

Verschwindet die Funktion p in diesem Beispiel an Gitterpunkten t_i, so ist wegen $a_i = 2$ die entstehende Koeffizientenmatrix in System (68) nicht mehr strikt diagonal-dominant. Trotzdem lässt sich die Konvergenzaussage des Satzes 44 auch in diesem Fall gewinnen.

Definition 46 (Irreduzibilität, Diagonaldominanz) *$A \in \mathbb{C}^{n\times n}$ ist irreduzibel diagonal-dominant, falls es folgende 2 Bedingungen erfüllt:*

(i) *A ist irreduzibel (nicht zerlegbar), d.h. es existiert keine Permutationsmatrix $P \in \mathbb{R}^{n\times n}$, mit der A in die Blockform*

$$PAP^T \;=\; \begin{pmatrix} A_{11} & A_{12} \\ O & A_{22} \end{pmatrix}$$

mit quadratischen $(n_1, n_1)-$ und $(n_2, n_2)-$Matrizen A_{11} und A_{22}, $n_1 + n_2 = n$, und Nullmatrix O überführbar ist.

(ii) *A ist diagonal-dominant, d.h.*

$$\sum_{\substack{j=1\\ j\neq i}}^{n} |a_{ij}| \;\leq\; |a_{ii}|\,, \quad i = 1(1)n\,,$$

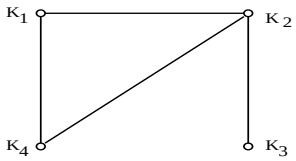
und mindestens eine Ungleichung ist mit $<$ erfüllt.

Um die Irreduzibilität einer Matrix A nachzuweisen, zeichnen wir am besten ihren Graphen.

Definition 47 (Graph einer Matrix) *Zu gegebener Matrix $A \in \mathbb{K}^{n\times n}$ und Indexmenge $I = \{1, 2, \ldots, n\}$ heisst die Menge $G(A) := \{(i, k) \in I \times I \mid a_{ik} \neq 0\}$ Graph der Matrix A.*

Bezeichnen wir mit $K_1, K_2, \ldots, K_n$ die *Knoten* des Graphen $G(A)$ in der Ebene, so können wir jedes Element $(i, k) \in G(A)$ darstellen, indem wir den Knoten K_i mit K_k durch eine Linie, auch *Kante* genannt, verbinden. Die Kanten (i, i) werden nicht eingezeichnet. Z.B. ergibt sich für die 4-reihige Matrix

$$A = \begin{pmatrix} 2 & 4 & 0 & 3 \\ 1 & 0 & 7 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 4 & 0 & 1 \end{pmatrix}$$



der nebenstehende zusammenhängende Matrixgraph. In [12], S. 145, wird dann gezeigt, dass A irreduzibel ist:

Satz 48 *Eine Matrix A ist genau dann irreduzibel, wenn ihr Graph $G(A)$ zusammenhängt, d.h. 2 beliebige Knoten stets durch eine Kantenfolge verbunden sind.*

Beispiel 49 Das vereinfachte Randwertproblem (vgl. Beispiel 45)

$$-\frac{d^2x}{dt^2} \;=\; q(t)\,, \quad x(a) = x_a,\; x(b) = x_b \tag{69}$$

auf dem Intervall $I = [a, b]$ führt analog zu Beispiel 45 auf das tridiagonale Gleichungssystem

$$\begin{array}{rrrcll} 2x_1 & - x_2 & & = & b_1 + x_a\,, & i = 1 \\ - x_{i-1} & + 2x_i & - x_{i+1} & = & b_i\,, & i = 2(1)n-2 \\ & - x_{n-2} & + 2x_{n-1} & = & b_{n-1} + x_e\,, & i = n-1\,, \end{array}$$

mit diagonal-dominanter Matrix A. Wegen $a_{i,i+1} = -1$ ist ihr Graph $G(A)$ offensichtlich zusammenhängend, womit die Irreduzibilität folgt. □

Für irreduzibel diagonal-dominante Matrizen wird in [12], S. 161, die Konvergenz der beiden Iterationsverfahren nachgewiesen:

Satz 50 *Ist die Koeffizientenmatrix A irreduzibel diagonal-dominant, so ist A regulär und das lineare System eindeutig lösbar. Gesamtschritt- und Einzelschritt-Verfahren konvergieren mit beliebiger Startnäherung x_0 gegen die Lösung x_*.*

Beispiel 51 Wir betrachten nunmehr ein praxisbezogenes 2-dimensionales Modell. Sei $\Omega = (0,1) \times (0,1)$ das Einheitsquadrat der Ebene und $f : \Omega \subset \mathbb{R}^2 \to \mathbb{R}$ eine auf Ω definierte reelle Funktion, die z.B. eine Ladungsdichte einer quadratischen Platte beschreibt. Gesucht ist das elektrische Potenzial $u(x,y)$ in jedem Punkt $(x,y) \in \Omega$. Man erhält es als Lösung einer partiellen Differenzialgleichung 2.Ordnung, der so genannten *Poisson-Gleichung*

$$\begin{array}{rcll} -\dfrac{\partial^2 u}{\partial x^2} - \dfrac{\partial^2 u}{\partial y^2} & = & f(x,y) & , \quad (x,y) \in \Omega \\ u(x,y) & = & \phi(x,y) & , \quad (x,y) \in \Gamma = \partial\Omega\,. \end{array}$$

Das Potenzial $\phi(x,y)$ auf dem Rand $\partial\Omega$ des Gebietes wird vorgegeben. Wir diskretisieren das Problem mit der Gitterweite $h = 1/N, N \in \mathbb{N}$ auf dem Gitter

$$\Omega_h = \{(x,y) \mid (x,y) = (ih, jh),\ \ i,j = 1,2,\ldots,N-1\}\,,$$

indem wir die partiellen Ableitungen durch Differenzenquotienten approximieren. Mit den Abkürzungen $u_{ij} := u(ih,jh),\ f_{ij} := f(ih,jh)$ erhalten wir

$$-\frac{1}{h^2}(u_{i-1,j} - 2u_{ij} + u_{i+1,j}) - \frac{1}{h^2}(u_{i,j-1} - 2u_{ij} + u_{i,j+1}) \;=\; f_{ij}\,, \quad i,j = 1,2,\ldots,N-1\,,$$

woraus sich die so genannte Fünfpunkteformel

$$\frac{1}{h^2}(4u_{ij} - u_{i-1,j} - u_{i+1,j} - u_{i,j-1} - u_{i,j+1}) \;=\; f_{ij}\,,\ \ i,j = 1,2,\ldots,N-1\,,$$

ergibt. Zusammen mit den Randwerten $u_{ij} = \phi(ih,jh)\,,\ i = 0,N$ oder $j = 0,N$ entsteht ein großdimensionales lineares Gleichungssystem für die $(N-1)^2$ unbekannten Werte u_{ij} des Potenzials. Zeilenweise (lexikografische) Anordnung der Unbekannten $x = (x_1, x_2, ..., x_n)^T$, $x \in \mathbb{R}^n\,,\ n = (N-1)^2$ mit

$$x_k = u_{ij}\,, \quad k = i + (j-1)(N-1)\,, \quad 1 \le i,j \le N$$

liefert ein reelles lineares Gleichungssystem $Ax = a$, das eine blocktridiagonale Koeffizientenmatrix

$$A = \frac{1}{h^2}\begin{pmatrix} T & -I & & \cdots & O & O \\ -I & T & -I & \cdots & O & O \\ O & -I & T & \cdots & O & O \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ O & O & O & \cdots & T & -I \\ O & O & O & \cdots & -I & T \end{pmatrix}$$

mit den $(N-1)(N-1)$ - Blöcken

$$T = \begin{pmatrix} 4 & -1 & & \cdots & 0 & 0 \\ -1 & 4 & -1 & \cdots & 0 & 0 \\ 0 & -1 & 4 & \cdots & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & 4 & -1 \\ 0 & 0 & 0 & \cdots & -1 & 4 \end{pmatrix}, \quad T \in \mathbb{R}^{(N-1)^2}$$

und den $(N-1)(N-1)$ -Einheitsmatrizen I besitzt. Die Besetztheitsstruktur von A kann in MATLAB mittels des Kommandos `spy(A)` visualisiert werden. Die Fälle $N = 6$ und $N = 11$ sind in Abb. 11 und 12 dargestellt. Bei einer Anzahl von $N = 101$ Teilintervallen pro

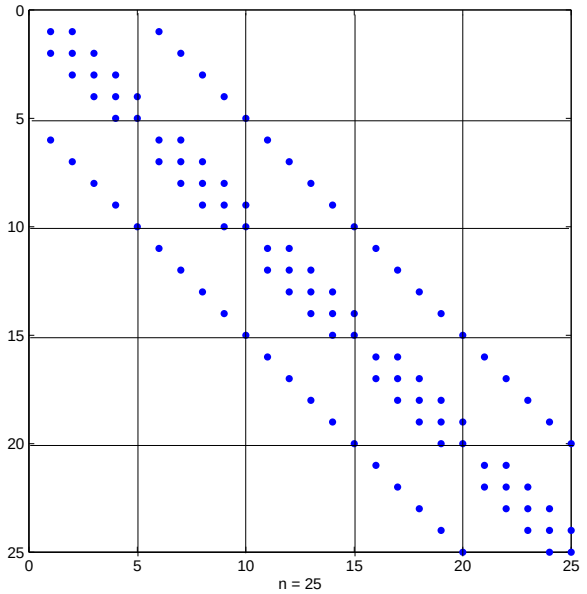


Abbildung 11: $n = 25$

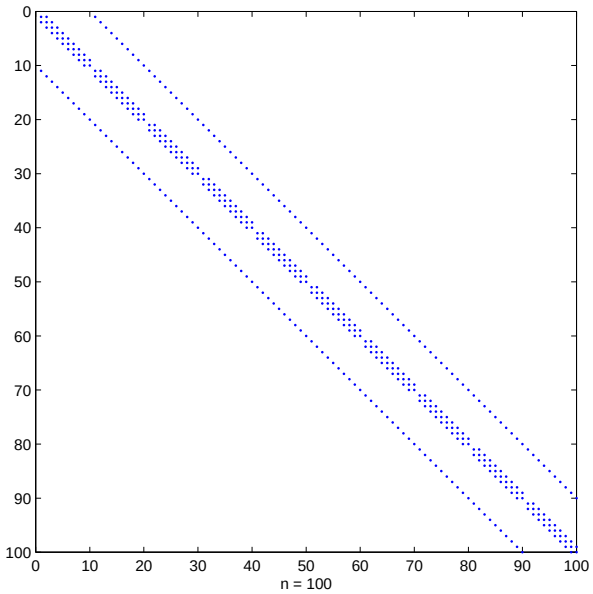


Abbildung 12: $n = 100$

Dimension ergeben sich $n = (N-1)^2 = 10\,000$ Unbekannte und A ist somit eine $10\,000 \times 10\,000$-Matrix! An der Gesamtstruktur dieser schwach besetzten Matrix (engl. *sparse matrix*) A überprüft man die Diagonaldominanz und die Irreduzibilität. Nach Satz 50 folgt daraus die eindeutige Lösbarkeit und die Konvergenz von Gesamtschritt- und Einzelschritt-Verfahren. Zudem lässt sich in beiden Verfahren der Spektralradius der Iterationsmatrix (vgl. [32]) durch $\varrho(B) = 1 - |\mathcal{O}(h^2)|$ abschätzen, was bei genauer Lösungsapproximation eine sehr

geringe Konvergenzgeschwindigkeit bedeutet.

$$A = \frac{1}{h^2}\begin{pmatrix}
\begin{matrix} 4 & -1 & \cdots & 0 \\ -1 & 4 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 4 \end{matrix} & \begin{matrix} -1 & 0 & \cdots & 0 \\ 0 & -1 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & -1 \end{matrix} & \cdots & O \\
\begin{matrix} -1 & 0 & \cdots & 0 \\ 0 & -1 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & -1 \end{matrix} & \begin{matrix} 4 & -1 & \cdots & 0 \\ -1 & 4 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 4 \end{matrix} & \cdots & O \\
\cdots & \cdots & \cdots & \cdots \\
O & O & \cdots & \begin{matrix} 4 & -1 & \cdots & 0 \\ -1 & 4 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 4 \end{matrix}
\end{pmatrix}$$

N	$n = (N-1)^2$	$\varrho(B)$	k
5	16	0.8090	29
11	100	0.9595	34
21	400	0.9888	261
31	900	0.9949	468
41	1600	0.9971	565
51	2500	0.9981	1140

Tabelle 2: Konvergenz des Gesamtschritt-Verfahrens

Dies bestätigt Tabelle 2, in der der Spektralradius $\varrho(B)$ der Iterationsmatrix für wachsende Gleichungsanzahl n dargestellt wird und die Zahl k der benötigten Gesamtschritte des Jacobi-Verfahrens, um eine Lösungsgenauigkeit von 10^{-4} zu erhalten. Das Ergebnis der aufwändigen Rechnung mit $N = 31$ Teilpunkten bei 2 vorgegebenen Punktladungen $f(x, y)$ zeigt die Darstellung von u_{ij} auf dem Gitter Ω_h in Abb. 13 anschaulich. □

Relaxationsverfahren Die für großdimensionale Systeme oft typische geringe Konvergenzgeschwindigkeit kann durch so genannte *Relaxationsverfahren* als Verallgemeinerungen der beiden Verfahrensklassen verbessert werden. Mit sukzessiver Überrelaxation auf Basis des Jacobi-Verfahrens (JOR) bzw. des Gauß-Seidel-Verfahrens (SOR) lassen sich so Konvergenzbeschleunigungen bei speziell strukturierten Gleichungssystemen erreichen.

Ausgehend vom k-ten Näherungsvektor x_k liefert das Basisverfahren (54) den Vektor $\xi_k = Bx_k + b$. Ein Relaxationsverfahren bildet nun eine Linearkombination der beiden Vektoren

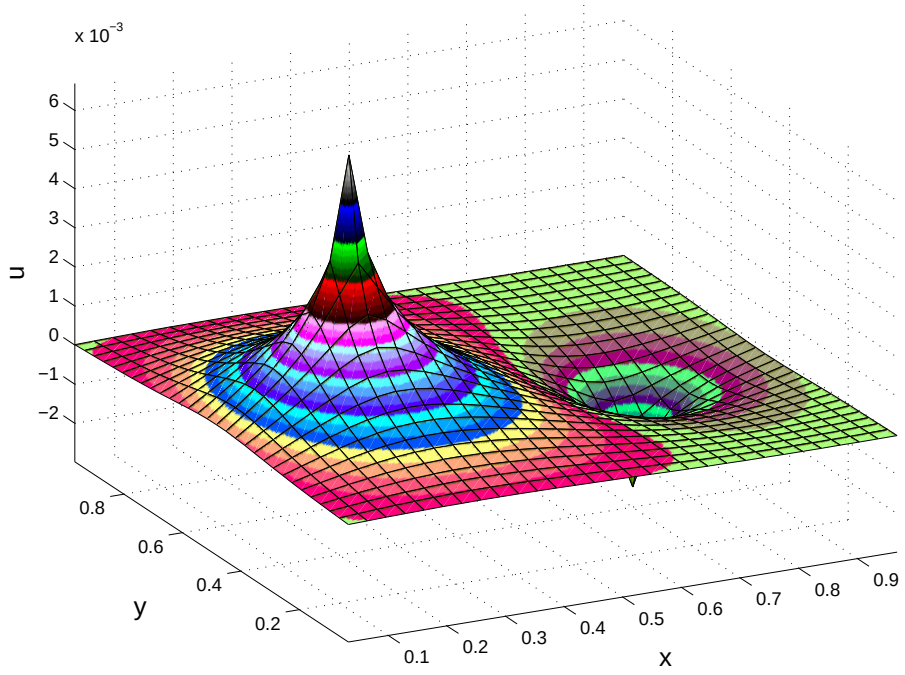


Abbildung 13: Näherungslösung von $u(x, y)$ mit $N = 31$

der Form

$$x_{k+1} \;=\; \omega\xi_k + (1-\omega)x_k \;=\; \omega(Bx_k + b) + (1-\omega)x_k \tag{70}$$

mit dem reellen Parameter $\omega > 0$ gemäß Abb. 14.

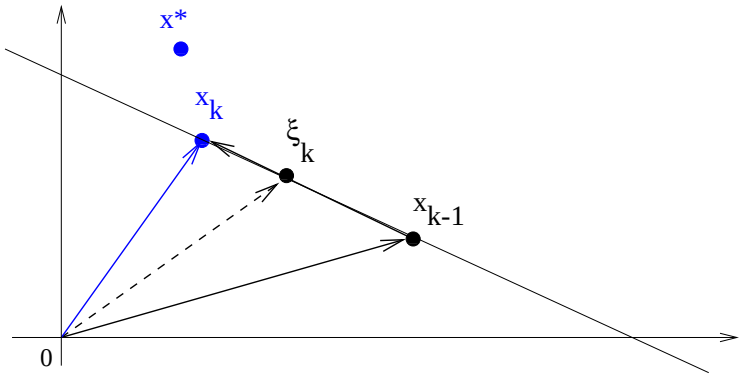


Abbildung 14: Überrelaxation mit $\omega > 1$

Das Basisverfahren ist mit $\omega = 1$ als Spezialfall enthalten. Während für $0 < \omega < 1$ *Unterrelaxation* vorliegt, spricht man bei $\omega > 1$ von *Überrelaxation*. Wir notieren das Verfahren in der Form

$$x_{k+1} \;=\; [(1-\omega)I + \omega B]x_k + \omega b\,, \quad k = 0, 1, 2, \ldots \tag{71}$$

und erhalten die parameterabhängige Iterationsmatrix $B_\omega = (1-\omega)I + \omega B$ mit dem *Relaxationsfaktor* ω. Eine optimale Wahl des Relaxationsfaktors mit dem Ziel, den Spektralradius

$\varrho(B_\omega)$ zu minimieren, hängt stark von den Eigenschaften von B ab. Eine Konvergenzbeschleunigung gegenüber dem Basisverfahren wird in der Regel durch Überrelaxation erreicht. Speziell ergibt sich Jacobi-Überrelaxation (JOR, engl. *Jacobi overrelaxation*) durch den Verfahrensschritt

$$x_{k+1} = D^{-1}[(1-\omega)D - \omega(L+R)]x_k + \omega D^{-1}a\ ,\ k = 0, 1, 2, \ldots \tag{72}$$

mit der Iterationsmatrix $B_\omega = D^{-1}[(1-\omega)D - \omega(L+R)]$. Die häufiger benutzte Gauß-Seidel-Überrelaxation (SOR, engl. *successive overrelaxation*) hat den Verfahrensschritt

$$x_{k+1} = (D+\omega L)^{-1}[(1-\omega)D - \omega R]x_k + \omega(D+\omega L)^{-1}a\ ,\ k = 0, 1, 2, \ldots \tag{73}$$

mit der Iterationsmatrix $B_\omega = (D+\omega L)^{-1}[(1-\omega)D - \omega R]$. Die Konvergenzanalyse von JOR und SOR ist komplizierter als die der beiden Basisverfahren. Für hermitesch positiv definite Koeffizientenmatrizen gilt der Konvergenzsatz von YOUNG:

Satz 52 (D.M. Young) *Ist A hermitesch und positiv definit (hpd), so konvergiert das Gauß-Seidel-Relaxationsverfahren (SOR) für jede Startnäherung x_0 , falls $0 < \omega < 2$ gilt.*

Den Beweis und eine detaillierte und umfangreiche Darstellung weiterer Eigenschaften von Relaxationsverfahren findet man in [12], S.134f. Die erzeugenden Matrizen N^{-1} gemäß (52) und Iterationsmatrizen lauten

Verfahren	Erz. Matrix N^{-1}	Iterationsmatrix B_ω
Jacobi	D^{-1}	$-D^{-1}(L+R)$
Gauß-Seidel	$(D+L)^{-1}$	$-(D+L)^{-1}R$
JOR	ωD^{-1}	$D^{-1}[(1-\omega)D - \omega(L+R)]$
SOR	$\omega(D+\omega L)^{-1}$	$(D+\omega L)^{-1}[(1-\omega)D - \omega R]$

Die Bedeutung von Relaxationsverfahren als selbstständige Verfahren ist relativ gering. Allerdings werden sie oft als Teilkomponenten in leistungsfähigere Zugänge eingebaut, z.B. zur Vorkonditionierung von Projektionsverfahren und als Bestandteile (sog. Glättungs-Verfahren) von Mehrgitterverfahren.

2.3 Krylov-Unterräume und Arnoldi-Verfahren

Wir betrachten in diesem Abschnitt großdimensionale reelle Gleichungssysteme

$$Ax \ = \ a\ , \quad A \in \mathbb{R}^{n\times n},\ a, x \in \mathbb{R}^n \tag{74}$$

mit regulärer Koeffizientenmatrix A. Weitere einschränkende Eigenschaften, wie Symmetrie oder Definitheit, sollen dagegen nicht vorausgesetzt werden. Ist eine Näherungslösung x_m bekannt, so heißt

$$r_m \ := \ a - Ax_m \tag{75}$$

Residuenvektor (das Residuum) von x_m. Das Ziel iterativer Verfahren besteht darin, ausgehend von einem Startvektor x_0, eine Folge von Näherungen (x_m) zu erzeugen, deren Residuen

r_m rasch gegen Null konvergieren. Ein Projektionsverfahren sucht eine Näherungslösung x_m in einem m-dimensionalen affinen Unterraum $x_0 + \mathcal{K}_m$ von $\mathbb{R}^n$. Dabei soll m sehr klein gegenüber der Dimension n sein und x_0 eine geeignete Startnäherung darstellen. Um ein Element δ_m des „Suchraumes“ (engl. *search subspace*) $\mathcal{K}_m$ bestimmen zu können, sind m Bedingungen aufzustellen, die in der Regel als Orthogonalitätsbedingungen formuliert werden. Dazu definiert man einen m-dimensionalen Test-Unterraum (engl. *subspace of constraints*) $\mathcal{L}_m$ und fordert, dass das Residuum r_m orthogonal auf $\mathcal{L}_m$ steht. Es ist also ein solches $x_m \in x_0 + \mathcal{K}_m$ zu finden, für das gilt

$$r_m \;=\; a - Ax_m \;\perp\; \mathcal{L}_m\,. \tag{76}$$

Wegen $x_m = x_0 + \delta_m\,, \delta_m \in \mathcal{K}_m\,,$ kann die Approximationsaufgabe auch durch

$$\langle r_0 - A\delta_m, w\rangle_2 \;=\; 0 \qquad \forall\, w \in \mathcal{L}_m \tag{77}$$

mit dem Euklidischen Skalarprodukt in $\mathbb{R}^n$ beschrieben werden (vgl. Abb. 15).

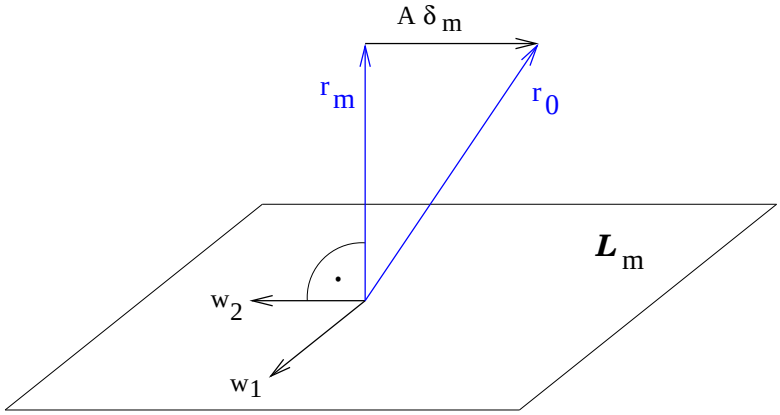


Abbildung 15: Projektion des Residuums auf $\mathcal{L}_m$

Definition 53 (Projektionsverfahren)

(*i*) *Ein Projektionsverfahren bestimmt zu $x_0 \in \mathbb{R}^n$ eine Folge von Näherungslösungen $x_m = x_0 + \delta_m\,,\ \delta_m \in \mathcal{K}_m\,,$ mit den m-dimensionalen Unterräumen $\mathcal{K}_m$ und $\mathcal{L}_m\,,$ die die Bedingung (76) erfüllen.*

(*ii*) *Ist $\mathcal{K}_m = \mathcal{L}_m\,,$ so heißt (76) Galerkin-Bedingung und das Projektionsverfahren ist orthogonal.*

(*iii*) *Ist $\mathcal{K}_m \neq \mathcal{L}_m\,,$ so heißt (76) Petrov-Galerkin-Bedingung und das Projektionsverfahren ist schief.*

Sei $V = (v_1, v_2, \ldots, v_m)$ die $n\times m-$Matrix, deren Spaltenvektoren eine Basis des Unterraumes $\mathcal{K}_m$ bilden und $W = (w_1, w_2, \ldots, w_m)$ eine entsprechende Matrix für eine Basis von $\mathcal{L}_m\,.$ Dann kann x_m offensichtlich in der Gestalt

$$x_m \;=\; x_0 + Vy \quad \text{mit} \quad y \in \mathbb{R}^m \tag{78}$$

dargestellt werden und die Orthogonalitätsbedingung (76) lässt sich in der Form

$$W^T r_m \;=\; W^T(a - Ax_m) \;=\; W^T(r_0 - AVy) \;=\; 0$$

notieren. Mit den Abkürzungen $A_m := W^T AV$ und $a_m := W^T r_0$ reduziert sich das großdimensionale System $Ax = a$ so auf das i.A. kleine m-dimensionale Gleichungssystem

$$A_m y = a_m$$

dessen Lösung y mit geringem Aufwand bestimmbar ist, wenn wir die Regularität von A_m voraussetzen. Formale Auflösung nach y und Einsetzen in (78) liefert dann die kompakte Lösungsdarstellung

$$x_m = x_0 + V(W^T AV)^{-1} W^T r_0 \,. \tag{79}$$

Alternativ zur Orthogonalitätsforderung (76) kann man auch eine Minimierungsbedingung

$$\|r_m\|_2 = \|a - Ax_m\|_2 \implies Min! \quad \text{für} \quad x_m \in x_0 + \mathcal{K}_m \tag{80}$$

mit der Euklidischen Norm für das Residuum aufstellen. Setzen wir die Darstellung (78) für x_m ein, so ergibt sich ein freies Minimierungsproblem. Die notwendige Bedingung für ein lokales Extremum liefert die Gleichungen

$$(AV)^T (r_0 - AVy) = 0 \,.$$

Wir setzen also speziell den Unterraum $\mathcal{L}_m = A\mathcal{K}_m$ an, so dass wir $W = AV$ wählen können. Dann hat AV den Rang m, womit $A_m = (AV)^T(AV)$ regulär ist und diese Bedingung auch als Projektionsverfahren in Definition 53 eingeordnet werden kann. Eine detaillierte Darstellung zahlreicher Projektionsverfahren findet man in [27]. Mit den ausgewählten 2 Basisverfahren GMRES und BiCG verdeutlichen wir wichtige Grundprinzipien und orientieren uns dabei stark an der Darstellung in [27].

Krylov-Unterräume Eine effiziente Bestimmung der Lösungsnäherungen x_m erfordert einen schrittweisen Aufbau der Unterräume $\mathcal{K}_m$ für $m = 1, 2, 3, \ldots$ mit möglichst wenigen Operationen, z.B. mit wenigen Matrix-Vektor-Multiplikationen Av. Dafür haben sich *Krylov-Unterräume* bewährt.

Definition 54 (Krylov-Unterraum) *Zu gegebenem $v \in \mathbb{R}^n$ hat ein Krylov-Unterraum die Form*

$$\mathcal{K}_m = \mathcal{K}_m(A, v) := span\{v, Av, \ldots, A^{m-1}v\} \,. \tag{81}$$

Speziell sei zu einer Lösungsnäherung x_0 der Krylov-Unterraum

$$\mathcal{K}_m = \mathcal{K}_m(A, r_0) := span\{r_0, Ar_0, \ldots, A^{m-1}r_0\} \tag{82}$$

mit dem Residuum $r_0 = b - Ax_0$ definiert.

Offenbar ist $\mathcal{K}_m$ der Unterraum aller Vektoren in $\mathbb{R}^n$, die man als $x = p(A)v$ darstellen kann, wobei p ein Polynom vom maximalen Grad $m-1$ ist. Als *Minimalpolynom* eines gegebenen Vektors v bezeichnet man das Nichtnullpolynom p mit minimalem Grad, mit dem $p(A)v = 0$ gilt. Der Grad μ dieses Polynoms von v in Bezug auf A wird *Grad von v (bezüglich A)* genannt. Als Folgerung aus dem bekannten Cayley-Hamilton-Theorem kann der Grad von v die Raumdimension n nicht übersteigen. Die folgenden Eigenschaften von Krylov-Unterräumen sind nun leicht zu beweisen:

Satz 55

(*i*) *Sei μ der Grad von v. Dann ist $\mathcal{K}_\mu$ invariant unter A und $\mathcal{K}_m = \mathcal{K}_\mu \ \forall m \geq \mu$.*

(*ii*) *Der Krylov-Unterraum $\mathcal{K}_m$ hat genau dann die Dimension m, wenn der Grad μ von v nicht kleiner als m ist, d.h.*

$$dim(\mathcal{K}_m) = m \quad \Longleftrightarrow \quad grad(v) \geq m. \tag{83}$$

Damit gilt $dim(\mathcal{K}_m) = \min\{m, grad(v)\}$.

BEWEIS: Die Vektoren $v, Av, \ldots, A^{m-1}v$ bilden genau dann eine Basis von $\mathcal{K}_m$, wenn für jede Menge von m Skalaren α_i, $i = 0, \ldots, m-1$ unter denen mindestens ein $\alpha_i \neq 0$ ist, die Linearkombination $\sum_{i=0}^{m-1} \alpha_i A^i v \neq 0$ ist. Dies ist äquivalent der Bedingung, dass das einzige Polynom mit Grad $\leq m-1$, für dass $p(A)v = 0$ gilt, das Nullpolynom ist. □

Zuerst wollen wir eine Orthonormalbasis des Krylov-Raumes $\mathcal{K}_m$ berechnen. Dazu wenden wir das aus der Linearen Algebra bekannte Gram-Schmidt-Orthogonalisierungsverfahren auf die Krylov-Basis $\{v, Av, \ldots, A^{m-1}v\}$ an und erhalten das *Arnoldi-Verfahren* in Algorithmus 56. Die Vektoren v_i, $i = 1, \ldots, m$ sind per Konstruktion orthonormal. Dass sie auch den

Algorithmus 56 (Arnoldi-Verfahren)

Function $[V, H] =$ Arnoldi(v, A, m)

1. Normiere $v_1 = v/\|v\|_2$
2. For $j = 1(1)m$ do
   1. Berechne $h_{ij} = (Av_j, v_i)$ für $i = 1(1)j$
   2. Berechne $w_j = Av_j - \sum_{i=1}^{j} h_{ij} v_i$
   3. Berechne $h_{j+1,j} = \|w_j\|_2$
   4. Falls $h_{j+1,j} = 0$, so STOP
   5. Normiere $v_{j+1} = w_j / h_{j+1,j}$
3. Return $V = (v_j)$ und $H = (h_{i,j})$

Raum $\mathcal{K}_m$ aufspannen, folgt aus der Tatsache, dass jeder Vektor v_j der Form $q_{j-1}(A)v_1$ genügt, wobei q_{j-1} ein Polynom vom Grade $j-1$ ist. Denn für $j = 1$ ist das Ergebnis offensichtlich, da $v_1 = q_0(A)v_1$ mit $q_0(A) \equiv 1$. Angenommen, das Ergebnis gilt für alle $n \leq j$, d.h. $v_n = q_{n-1}(A)v_1$. Für $j+1$ erhält man dann

$$h_{j+1}v_{j+1} = Av_j - \sum_{i=1}^{j} h_{ij} v_i = Aq_{j-1}(A)v_1 - \sum_{i=1}^{j} h_{ij} q_{i-1}(A)v_1 \,,$$

woraus sich die Darstellung des Polynoms

$$q_j(A) = \underbrace{Aq_{j-1}(A)}_{Grad\ j} - \underbrace{\sum_{i=1}^{j} h_{ij} q_{i-1}(A)}_{Grad\ j-1}$$

ergibt. Also kann der Vektor v_{j+1} durch ein Polynom der Form $q_j(A)v_1$ dargestellt werden, wobei q_j vom Grad j ist. Induktiv haben wir damit das Verfahren verifiziert:

Satz 57 *Angenommen, Algorithmus 56 stoppt nicht vor dem m-ten Schritt. Dann bilden die Vektoren $v_1, v_2, \ldots, v_m$ eine Orthonormalbasis des Krylov-Unterraums $\mathcal{K}_m(A, v)$.*

Für die folgenden Verfahren ist die Matrixdarstellung des Algorithmus wesentlich.

Satz 58 *V_m sei die $n \times m$-Matrix mit den Spaltenvektoren $v_1, \ldots, v_m$. Weiterhin sei $\bar{H}_m$ die $(m+1) \times m$-Hessenberg-Matrix, deren Einträge $h_{ij} \neq 0$ durch den Algorithmus 56 definiert sind und H_m die Matrix, die entsteht, wenn man in der Matrix $\bar{H}_m$ die letzte Zeile weglässt. Dann gelten die folgenden Gleichungen:*

$$AV_m = V_{m+1}\bar{H}_m = V_m H_m + w_m e_m^T \tag{84}$$

$$V_m^T A V_m = H_m\,. \tag{85}$$

Beweis: Aus den Schritten 2.2, 2.3 und 2.5 des Algorithmus 56 folgt die Gleichung

$$Av_j = \sum_{i=1}^{j+1} h_{ij} v_i, \qquad j = 1, 2, \ldots, m, \tag{86}$$

die in Matrixdarstellung mit dem m-ten Einheitsvektor e_m die Beziehung (84) liefert. Gleichung (85) erhält man, wenn man beide Seiten von (84) mit V_m^T multipliziert und die Orthonormalität von $\{v_1, \ldots, v_m\}$ berücksichtigt. □

Bemerkung 59 1. Der Algorithmus stoppt, wenn die Norm von w_j im Schritt j verschwindet und so v_{j+1} nicht berechnet werden kann. In [27] wird gezeigt, dass das Arnoldi-Verfahren genau dann im Schritt j abbricht, wenn das Minimalpolynom von v_1 den Grad j hat.

2. Bei größerer Dimension m empfiehlt sich die Anwendung der *modifizierten Gram-Schmidt-Orthogonalisierung*, welche auf die Variante des Arnoldi-Verfahrens in Algorithmus 60 führt.

Noch robustere Varianten des Arnoldi-Verfahrens, z.B. unter Benutzung der Householder-Orthogonalisierung, findet man in [27].

2.4 GMRES-Verfahren und BiCG-Verfahren

Das GMRES-Verfahren Das *GMRES-Verfahren (Generalized Minimum Residual Method)* ist ein Projektionsverfahren, bei dem $\mathcal{K}_m = \mathcal{K}_m(A, v_1)$ und $\mathcal{L}_m = A\mathcal{K}_m$ gewählt wird. Dabei ist $\mathcal{K}_m$ der m-te Krylov-Unterraum mit $v_1 = r_0/\|r_0\|_2$. Diese Technik minimiert zugleich die Residualnorm über alle Vektoren in $x_0 + \mathcal{K}_m$, weshalb wir von Ansatz (80) ausgehen wollen. Jeder Vektor x_m aus $x_0 + \mathcal{K}_m$ kann als

$$x_m = x_0 + V_m y \tag{87}$$

Algorithmus 60 (Arnoldi-Verfahren modifiziert)
Function $[V, H] = \text{Arnoldimod}(v, A, m)$

1. Normiere $v_1 = v/\|v\|_2$
2. For $j = 1(1)m$ do
   1. Berechne $w_j = Av_j$
   2. For $i = 1(1)j$ do
      1. $h_{ij} = (w_j, v_i)$
      2. $w_j = w_j - h_{ij}v_i$
   3. Berechne $h_{j+1,j} = \|w_j\|_2$
   4. Falls $h_{j+1,j} = 0$, so STOP
   5. Normiere $v_{j+1} = w_j/h_{j+1,j}$
3. Return $V = (v_j)$ und $H = (h_{i,j})$

notiert werden, wobei y ein m-dimensionaler Vektor ist. Definieren wir die Norm des Residuums der Näherung x_m zu

$$J(y) \;=\; \|b - Ax_m\|_2 = \|b - A(x_0 + V_m y)\|_2 \,,$$

so ergibt sich aus Gleichung (84) mit der Abkürzung $\beta = \|r_0\|_2$

$$\begin{aligned} J_m(y) \;&=\; \|r_0 - AV_m y\|_2 \;=\; \|\beta v_1 - V_{m+1}\bar{H}_m y\|_2 \\ &=\; \|V_{m+1}(\beta e_1 - \bar{H}_m y)\|_2 \;=\; \|\beta e_1 - \bar{H}_m y\|_2 \,. \end{aligned}$$

Die letzte Umformung erhalten wir, da die Spaltenvektoren von V_{m+1} orthonormal sind. Als GMRES-Näherung wird derjenige Vektor $x_m \in x_0 + \mathcal{K}_m$ bestimmt, der die Residualnorm $J(y)$ minimiert. Das bedeutet jedoch die Lösung der Minimierungsaufgabe

$$J(y_m) \;=\; \min_{y\in\mathbb{R}^m} J(y) \;=\; \min_{y\in\mathbb{R}^m} \|\beta e_1 - \bar{H}_m y\|_2 \tag{88}$$

in $\mathbb{R}^m$. Die Berechnung dieses Minimierers y_m ist nicht aufwändig, da es nur der Lösung eines $(m+1)\times m$-dimensionalen Quadratmittel-Problems (engl. *least-squares*) bedarf, bei dem m im Allgemeinen klein ist. Nutzen wir die modifizierte Gram-Schmidt-Orthogonalisierung im Arnoldi-Schritt zur Bestimmung von V_m, so erhalten wir nun Algorithmus 61.

Bemerkung 62 1. Um das Minimierungsproblem $\min \|\beta e_1 - \bar{H}_m y\|_2$ zu lösen, sollte man die Hessenbergmatrix durch QR-Zerlegung mittels Householder-Matrizen in die obere Dreiecksform bringen.

2. Algorithmus 61 stoppt vorzeitig in Schritt 3.4., falls $h_{j+1,j} = 0$ gilt. Dann kann der nächste Arnoldi-Vektor v_{j+1} nicht berechnet werden. An diesem Punkt ist auch der Residuenvektor gleich Null, d.h. der Algorithmus liefert die exakte Lösung. Auch die Umkehrung ist wahr: Falls der Algorithmus im Schritt j mit $b - Ax_j = 0$ abbricht, ist $h_{j+1,j} = 0$.

Algorithmus 61 (GMRES-Verfahren)
Function $[x_m] = \text{gmres}(A, a, m, x_0)$

1. Berechne $r_0 = a - Ax_0$, $\beta = \|r_0\|_2$ und $v_1 = r_0/\beta$
2. Definiere die $(m+1) \times m$-Matrix $\bar{H}_m = (h_{ij})$. Setze $\bar{H}_m = 0$.
3. For $j = 1(1)m$ do
   - 3.1. Berechne $w_j = Av_j$
   - 3.2. For $i = 1(1)j$ do
     1. $h_{ij} = (w_j, v_i)$
     2. $w_j = w_j - h_{ij}v_i$
   - 3.3. Berechne $h_{j+1,j} = \|w_j\|_2$
   - 3.4. Falls $h_{j+1,j} = 0$, so setze $m = j$ und gehe zu 4.
   - 3.5. Normiere $v_{j+1} = w_j/h_{j+1,j}$
4. Berechne den Minimierer y_m nach (88)
5. Return $x_m = x_0 + V_m y_m$

Satz 63 *Sei A eine reguläre Matrix. Dann bricht der GMRES-Algorithmus 61 genau dann im j-ten Schritt mit $h_{j+1,j} = 0$ ab, wenn die Näherungslösung x_j die exakte Lösung ist.*

Ein wesentlicher Nachteil des GMRES-Verfahrens ist, dass Speicherplatzbedarf und Rechenaufwand mit m beträchtlich wachsen. Ein Ausweg besteht darin, den Algorithmus nach jeweils m Schritten mit $x_0 := x_m$ neu zu starten. Wir erhalten dann den GMRES(m)-Algorithmus 64 mit Restart. Das GMRES-Verfahren konvergiert theoretisch stets in maximal n Schritten. Ein Problem des GMRES(m)-Verfahrens mit Restart dagegen ist, dass das Verfahren im Allgemeinen stagnieren kann. Im Falle positiv definiter Matrizen A gilt jedoch (vgl. [27]) der

Satz 65 *Falls A eine symmetrische positiv definite Matrix ist, so konvergiert das Verfahren GMRES(m) für jedes $m \geq 1$.*

Bemerkung 66 1. Ist das System symmetrisch, so sind die erzeugten Hessenberg-Matrizen H_m tridiagonal. Damit stehen lediglich Dreiterm-Rekursionen im GMRES-Verfahren („kurze Rekursionen“), die wesentlich effizienter sind als die „langen Rekursionen“ des üblichen Verfahrens; ein Restart ist dann nicht erforderlich. Diese Form des Verfahrens heißt auch MINRES (engl. *minimal residual*).

2. Aus Effizienzgründen werden oft „unvollständige“ GMRES-Verfahren genutzt. Bei unvollständiger Orthogonalisierung in Algorithmusschritt 2 erhält man das so genannte *Quasi-*

Algorithmus 64 (GMRES(m)-Verfahren)
Function $[x_m] = \text{gmres\_restarted}(A, a, m, x_0, tol)$

1. Berechne $r_0 = a - Ax_0$, $\beta = \|r_0\|_2$ und $v_1 = r_0/\beta$
2. Bestimme die Arnoldi-Basis V_m sowie $\bar{H}_m$ mit Algorithmus 60
3. Berechne den Minimierer y_m nach (88)
4. Berechne $x_m = x_0 + V_m y_m$
5. Falls $\|a - Ax_m\|_2 < tol \cdot \|a\|_2$, so Return x_m
6. Setze $x_0 = x_m$ und gehe zu Schritt 1

GMRES-Verfahren (QGMRES). Zusammen mit einer genäherten Minimierung in Algorithmusschritt 3 ergibt sich das *direkte Quasi-GMRES-Verfahren (DQGMRES).*

3. Ist die Konvergenz von GMRES sehr langsam, so nimmt man häufig eine *Vorkonditionierung* vor. Darunter versteht man eine Transformation des Originalsystems $Ax = a$ in ein lineares System mit derselben Lösung, für das das GMRES-Verfahren jedoch schneller konvergiert. Angenommen, A sei regulär. Ein Vorkonditionierer (engl. *preconditioner*) ist eine reguläre Matrix $M \in \mathbb{R}^{n\times n}$, die die Matrix A in geeigneter Weise approximiert und zudem garantiert, dass lineare Systeme $Mx = a$ mit geringem Aufwand zu lösen sind. Bei *Links-Vorkonditionierung* erhält man so das System

$$M^{-1}Ax \;=\; M^{-1}a\,,$$

während durch Rechtsmultiplikation $AM^{-1}M$ eine *Rechts-Vorkonditionierung*

$$AM^{-1}y \;=\; a\,, \quad x = M^{-1}y$$

erreicht wird. Die geeignete Wahl des Vorkonditionierers M entscheidet oft wesentlich über die Konvergenzgeschwindigkeit des GMRES-Verfahrens.

Matlab stellt das GMRES-Verfahren ohne bzw. mit Restart bereit. Eine Vorkonditionierung ist möglich.

1. Mit dem Grundkommando `x = GMRES(A,b)` liefert das GMRES-Verfahren eine Näherungslösung des Systems $Ax = b$ mit dem Startvektor $x_0 = 0$ bei einer Standardgenauigkeit $tol = 10^{-6}$.
2. `x = GMRES(A,b,restart)` wendet GMRES(restart) mit jeweils `restart` Schritten und Standardgenauigkeit an, während die Toleranz im Aufruf
   `x = GMRES(A,b,restart,tol)` gesetzt werden kann.
3. Vorkonditionierte GMRES(restart)-Verfahren sind mit den Aufrufen
   `x = GMRES(A,b,restart,tol,maxit,M)` und
   `x = GMRES(A,b,restart,tol,maxit,M1,M2)`
   möglich. Dabei spezifiziert maxit die Maximalzahl auszuführender Iterationen und M oder $M = M_1M_2$ ist ein Links-Vorkonditionierer.

4. Die allgemeine Aufrufform des Kommandos GMRES
   `[x,flag,relres] = GMRES(A,b,restart,tol,maxit,M1,M2,x0)`
   akzeptiert eine selbstgewählte Startnäherung x_0 und liefert neben der Lösung x ein flag sowie das relative Residuum relres mit dem Wert $\|b - Ax\|_2/\|b\|_2$. Es ist flag $= 0$, falls das Verfahren mit relres $<$ tol innerhalb von maxit Iterationen konvergiert; andernfalls werden Fehlercodes geliefert.
5. Wird ein Parameter des Aufrufes nicht benötigt, so ist [ ] an seiner Stelle zu notieren.

N	$n = (N-1)^2$	t in sec	*iter*	*relres*
31	900	0.951	120	7.33e-13
41	1600	1.763	156	7.61e-13
51	2500	5.728	193	9.38e-13
61	3600	40.99	233	9.70e-13
71	4900	142.1	269	8.95e-13

Tabelle 3: Iterationen des GMRES-Verfahrens

Beispiel 67 1. Wir lösen das Poisson-Beispiel 51 mit dem GMRES-Verfahren bei einer vorgegebenen Genauigkeit $tol = 10^{-12}$ durch den Aufruf

```
[x,flag,relres,iter,resvec] = gmres(A,a,[],tol);
```

Dabei ist die Koefizientenmatrix A vom Typ `sparse`. Tabelle 3 zeigt die Entwicklung der Rechenzeit t und die Gesamtzahl *iter* der Iterationen, um das angegebene relative Residuum *relres* zu erreichen. Im Vergleich mit den Splitting-Verfahren der Tabelle 2 erkennt man den

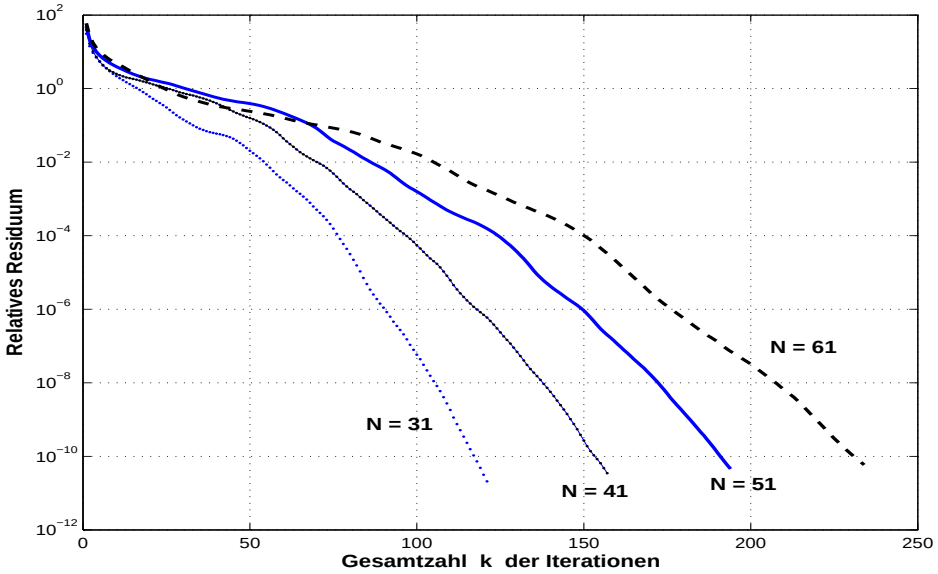


Abbildung 16: Konvergenz des GMRES-Verfahren

Vorteil des GMRES-Verfahrens. Die Abnahme des Residuums zeigt Abb. 16. Restarts sind nicht erforderlich.

2. Ein System $Ax = b$ mit unsymmetrischer Bandmatrix A wird durch die MATLAB-Kommandos

```
n = 2000; on = ones(n,1);
A = spdiags([-8*on 10*on -on],[-5,0,1],n,n);
A(n,1) = -10;  A(1,n) = 10; b = sum(A,2);
```

in schwach besetzter Speicherung erzeugt. Im Unterschied zum symmetrischen Poisson-System zeigt nun das Spektrum von A in Abb. 17 komplexe Eigenwerte. Algorithmus 64

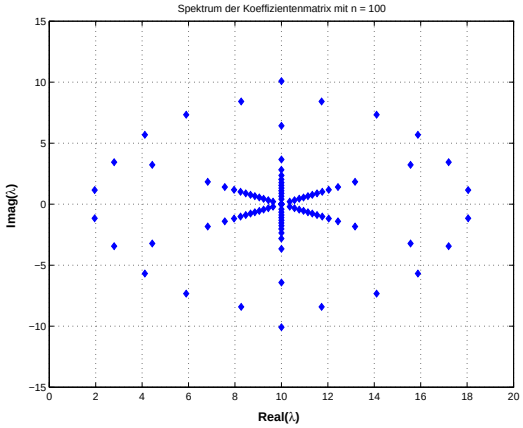


Abbildung 17: Spektrum von A

mit Restart nach jeweils m Iterationen ist sowohl im Falle $n = 2000$ (Abb. 18) als auch bei $n = 10\,000$ (Abb. 19) dem einfachen GMRES-Verfahren überlegen. □

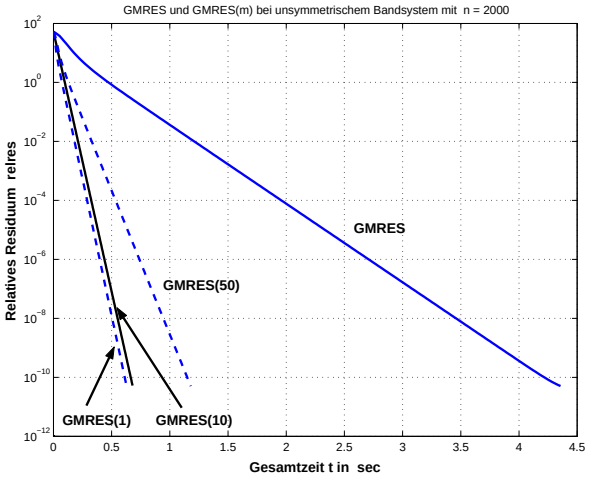


Abbildung 18: $n = 2000$

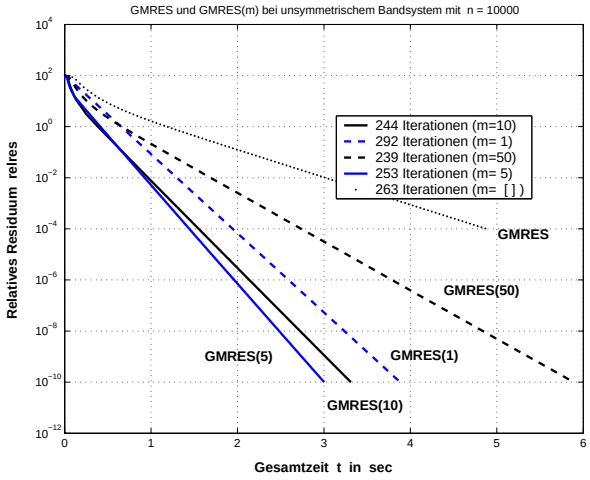


Abbildung 19: $n = 10000$

Das bikonjugierte Gradientenverfahren (BiCG) Für symmetrische positiv definite Systeme $Ax = a$ haben M.R.Hestenes und E.Stiefel die leistungsfähige Methode der konjugierten Gradienten (CG-Verfahren) entwickelt (vgl. [21]). Wir wollen den von C. Lanczos verallgemeinerten Zugang betrachten, mit dem beliebige reguläre Systeme gelöst werden können. Das so gewonnene *bikonjugierte Gradientenverfahren (engl. biconjucate gradient method (BiCG)* löst dabei nicht nur das Originalsystem $Ax = a$, sondern bei Bedarf auch duale (transponierte) Systeme $A^\top x^* = a^*$ mit beliebiger rechter Seite a^* . Es stellt ein

Projektionsverfahren auf den Krylov-Unterraum

$$\mathcal{K}_m := \mathcal{K}_m(A, v_1) = \text{span}\{v_1, Av_1, \ldots, A^{m-1}v_1\}\,, \tag{89}$$

dar, der orthogonal zum *transponierten Krylov-Unterraum*

$$\mathcal{L}_m := \mathcal{K}_m(A^\top, w_1) = \text{span}\{w_1, A^\top w_1, \ldots (A^\top)^{m-1} w_1\} \tag{90}$$

ist. Dabei sei $v_1 = r_0/\|r_0\|_2$. Der Vektor w_1 ist beliebig, vorausgesetzt $\langle v_1, w_1\rangle \neq 0$, er wird jedoch oft gleich v_1 gewählt. Falls auch das duale System $A^\top x^* = a^*$ gelöst werden soll, so erhält man w_1 analog zu v_1 durch die Normierung des Anfangsresiduums $r_0^* = a^* - A^\top x_0^*$ des dualen Systems.

Im ersten Teil des Verfahrens wird ein Paar biorthogonaler Basen $V = (v_1, v_2, \ldots, v_m)$ und $W = (w_1, w_2, \ldots, w_m)$ für die beiden Krylov-Unterräume $\mathcal{K}_m$ und $\mathcal{L}_m$ mittels der *Lanczos-Biorthogonalisierung* erzeugt (vgl. Algorithmus 68). Die berechneten Größen δ_{j+1} und β_{j+1}

Algorithmus 68 (Lanczos-Biorthogonalisierung)
Function $[V, W, T] =$ Biorthogonalisierung(A, m)

1. Wähle Vektoren v_1 und w_1 mit $(v_1, w_1) = 1$
2. Setze $\beta_1 = \delta_1 := 0$ und $w_0 = v_0 := 0$
3. For $j = 1(1)m$ do
   1. $\alpha_j = \langle Av_j, w_j\rangle$
   2. $\hat{v}_{j+1} = Av_j - \alpha_j v_j - \beta_j v_{j-1}$
   3. $\hat{w}_{j+1} = A^\top w_j - \alpha_j w_j - \delta_j w_{j-1}$
   4. $\delta_{j+1} = |\langle \hat{v}_{j+1}, \hat{w}_{j+1}\rangle|^{1/2}$. Falls $\delta_{j+1} = 0$ ist, so STOP.
   5. $\beta_{j+1} = \langle \hat{v}_{j+1}, \hat{w}_{j+1}\rangle/\delta_{j+1}$
   6. $w_{j+1} = \hat{w}_{j+1}/\beta_{j+1}$
   7. $v_{j+1} = \hat{v}_{j+1}/\delta_{j+1}$
4. Return $V = (v_j)$, $W = (w_j)$ und $T = tridiag(\delta_j, \alpha_j, \beta_j)$

stellen Skalierungsfaktoren für die Vektoren v_{j+1} und w_{j+1} dar. Aus den Schritten 5–7 des Algorithmus folgt, dass die Bedingung $\langle v_{j+1}, w_{j+1}\rangle = 1$ stets erfüllt ist. Die α_j, β_j und δ_j fassen wir zur Tridiagonalmatrix

$$T_m = \begin{pmatrix} \alpha_1 & \beta_2 & & & \\ \delta_2 & \alpha_2 & \beta_3 & & \\ & \cdot & \cdot & \cdot & \\ & & \delta_{m-1} & \alpha_{m-1} & \beta_m \\ & & & \delta_m & \alpha_m \end{pmatrix} \tag{91}$$

zusammen. Wir können feststellen, dass für die Vektoren $v_i \in \mathcal{K}_m(A, v_1)$ ist, während $w_j \in \mathcal{K}_m(A^\top, w_1)$ gilt. Der gesamte Algorithmus wird durch folgenden Satz begründet:

Satz 69 *Falls der Algorithmus nicht vor dem m-ten Schritt abbricht, so bilden die Vektoren v_i, $i = 1, \ldots, m$ und w_j, $j = 1, \ldots, m$ ein biorthonormales System, d.h.*

$$\langle v_j, w_i \rangle = \delta_{ij} \qquad i = 1(1)m,\ j = 1(1)m, \quad (\delta_{ij} - \textit{Kronecker-Symbol}) \,. \tag{92}$$

Darüber hinaus ist $(v_i)_{i=1,2,\ldots,m}$ eine Basis von $\mathcal{K}_m(A, v_1)$ und $(w_i)_{i=1,2,\ldots,m}$ eine Basis von $\mathcal{K}_m(A^\top, w_1)$. Es gelten folgende Gleichungen:

$$\begin{aligned} AV_m &= V_m T_m + \delta_{m+1} v_{m+1} e_m^\top & (93)\\ A^\top W_m &= W_m T_m^\top + \beta_{m+1} w_{m+1} e_m^\top & (94)\\ W_m^\top A V_m &= T_m & (95) \end{aligned}$$

Beweis: Die Biorthogonalität der Vektoren v_i und w_i kann induktiv gezeigt werden. Es gilt $\langle v_1, w_1 \rangle = 1$ nach Schritt 1. Seien bereits die Vektoren $v_1, \ldots, v_j$ und $w_1, \ldots, w_j$ biorthogonal. Wir wollen nun zeigen, dass die Vektoren $v_1, \ldots, v_{j+1}$ und $w_1, \ldots, w_{j+1}$ biorthogonal sind. Zuerst weisen wir nach, dass $\langle v_{j+1}, w_i \rangle = 0$ für $i \leq j$ ist. Denn falls $i = j$, so gilt

$$\langle v_{j+1}, w_j \rangle = \delta_{j+1}^{-1} [\langle A v_j, w_j \rangle - \alpha_j \langle v_j, w_j \rangle - \beta_j \langle v_{j-1}, w_j \rangle].$$

Das letzte Skalarprodukt verschwindet aufgrund der Induktionsannahme. Die beiden anderen Terme fallen wegen der Definition von $\alpha_j = \langle A v_j, w_j \rangle$ und der Tatsache weg, dass $\langle v_j, w_j \rangle = 1$ ist. Betrachten wir also das Skalarprodukt $\langle v_{j+1}, w_i \rangle$ mit $i < j$,

$$\begin{aligned} \langle v_{j+1}, w_i \rangle &= \delta_{j+1}^{-1} [\langle A v_j, w_i \rangle - \alpha_j \langle v_j, w_i \rangle - \beta_j \langle v_{j-1}, w_i \rangle]\\ &= \delta_{j+1}^{-1} [\langle v_j, A^\top w_i \rangle - \beta_j \langle v_{j-1}, w_i \rangle]\\ &= \delta_{j+1}^{-1} [\langle v_j, \beta_{i+1} w_{i+1} + \alpha_i w_i + \delta_i w_{i-1} \rangle - \beta_j \langle v_{j-1}, w_i \rangle] \,. \end{aligned}$$

Für $i < j - 1$ verschwinden wegen der Induktionsannahme alle Skalarprodukte in der obigen Gleichung. Für den Fall $i = j - 1$ gilt

$$\begin{aligned} \langle v_{j+1}, w_{j-1} \rangle &= \delta_{j+1}^{-1} [\langle v_j, \beta_j w_j + \alpha_{j-1} w_{j-1} + \delta_{j-1} w_{j-2} \rangle - \beta_j \langle v_{j-1}, w_{j-1} \rangle]\\ &= \delta_{j+1}^{-1} [\beta_j \langle v_j, w_j \rangle - \beta_j \langle v_{j-1}, w_{j-1} \rangle]\\ &= 0 \,. \end{aligned}$$

Analog kann gezeigt werden, dass $\langle v_i, w_{j+1} \rangle = 0$ für $i \leq j$ ist. Zudem gilt $\langle v_{j+1}, w_{j+1} \rangle = 1$. Damit ist die Induktionsbehauptung bewiesen. Der zweite Teil des Satzes kann analog zu Satz 58 und den Gleichungen (84), (84) für das Arnoldi-Verfahren bewiesen werden. □

Mit den ermittelten Lanczos-Basen der Krylov-Räume $\mathcal{K}_m$ und $\mathcal{L}_m$ können wir nun das lineare Gleichungssystem und das duale System leicht lösen. Der *zweiseitige Lanczos-Algorithmus (Bi-Lanczos)* startet dazu speziell mit $v_1 = r_0 / \|r_0\|_2$ und $\beta = \|r_0\|_2$. Die Petrov-Galerkin-Bedingung (76) liefert in Matrixnotation für $x_m = x_0 + V_m y_m$ unter Benutzung von (95)

$$\begin{aligned} 0 &= W_m^T (a - A x_m) \;=\; W_m^T (r_0 - A V_m y_m)\\ &= W_m^T \beta v_1 - W_m^T A V_m y_m\\ &= \beta e_1 - T_m y_m \end{aligned}$$

woraus die Darstellung

$$y_m \;=\; T_m^{-1}\beta e_1\;, \quad e_1 \;-\; \text{Einheitsvektor}$$

mit der Tridiagonalmatrix T_m folgt. Das Verfahren kann so in Form des Algorithmus 70 notiert werden. Betrachten wir den Schritt 3 genauer. Wenn wir eine LU-Zerlegung der

Algorithmus 70 (Bi-Lanczos-Verfahren)

Function $[x_m] = \mathsf{Bi\_Lanczos}(A, a, m, x_0)$

1. Berechne $r_0 = a - Ax_0\;,\; \beta = \|r_0\|_2$ und $v_1 = r_0/\beta$
2. Bestimme die Lanczos-Basen V_m, W_m und die Tridiagonalmatrix T_m mit Algorithmus 68
3. Löse $T_m y_m \;=\; \beta e_1$ nach y_m auf
4. Return $x_m = x_0 + V_m y_m$

Tridiagonalmatrix in der Form $T_m \;=\; L_m R_m$ vornehmen, so lassen sich damit die beiden $n \times m-$Matrizen

$$P_m \;=\; V_m R_m^{-1} \;=\; (p_1, p_2, \ldots, p_m) \quad \text{und} \quad P_m^* \;=\; W_m L_m^{-1} \;=\; (p_1^*, p_2^*, \ldots, p_m^*)$$

mit den Spalten p_j und p_j^* bestimmen. Das *bikonjugierte Gradientenverfahren (BiCG)* konstruiert sukzessive diese Matrizen und die Näherungslösungen x_j und x_j^* für das Originalsystem $Ax = a$ und für das duale lineare System $A^T x^* = a^*$ samt den Residuen r_j und r_j^*. Ausgehend vom Bi-Lanczos-Verfahren liefert dieses Toleranz-gesteuerte Verfahren 71 die Lösungen und Residuen, bis eine gewünschte Genauigkeit erreicht wird (vgl. [27]).

Bemerkung 72 1. Ist A symmetrisch und positiv definit, so reduziert sich der Algorithmus auf das *konjugierte Gradientenverfahren (CG)*, das z.B. in [20], Band 1, S. 128ff, ausführlich behandelt wird.

2. In jedem Verfahrensschritt treten im Gegensatz zum GMRES-Verfahren nun 2 Matrix-Vektor-Multiplikationen Ap_j und $A^\top p_j^*$ auf; zudem ist mit $\langle r_j, r_j^* \rangle = 0$ ein unerwünschter Verfahrensabbruch möglich. Allerdings liefert das Verfahren neben x_j auch die Näherungslösung x_j^* des dualen Systems.

3. Weitere Verbesserungen, wie das CGS-Verfahren und das BiCGSTAB-Verfahren, vermeiden die Multiplikation mit der Transponierten $A^\top$ und zeigen mitunter schnellere Konvergenz. Unerwünschte Verfahrensabbrüche werden durch eine so genannte look-ahead-Strategie vermieden, die der interessierte Leser in (vgl. [27], [21]) findet.

Matlab liefert das BiCG-Verfahren in folgenden Varianten:

1. Der Aufruf `x = BICG(A,b)` versucht eine Näherungslösung des Systems mit dem Startvektor $x_0 = 0$ bei einer Standardgenauigkeit $tol = 10^{-6}$ und maximaler Iterationszahl maxit $= \min(n, 20)$. Toleranz und Iterationszahl können mittels des Aufrufes `x = BICG(A,b,tol,maxit)` gesetzt werden.

Algorithmus 71 (Bi-CG-Verfahren)
Function $[x_j, x_j^*] = \mathsf{BiCG}(A, a, a^*, x_0, x_0^*, tol)$

1. Berechne $r_0 := a - Ax_0$ und $r_0^* := a^* - A^\top x_0^*$
2. Setze $p_0 := r_0$ und $p_0^* := r_0^*$
3. For $j = 0, 1, 2, \ldots$ while $\|r_j\|_2 > tol \cdot \|a\|_2$ do
   1. $\alpha_j := \langle r_j, r_j^* \rangle / \langle Ap_j, p_j^* \rangle$
   2. $x_{j+1} := x_j + \alpha_j p_j$
   3. $x_{j+1}^* := x_j^* + \alpha_j p_j^*$
   4. $r_{j+1} := r_j - \alpha_j Ap_j$
   5. $r_{j+1}^* := r_j^* - \alpha_j A^\top p_j^*$
   6. $\beta_j := \langle r_{j+1}, r_{j+1}^* \rangle / \langle r_j, r_j^* \rangle$
   7. $p_{j+1} := r_{j+1} + \beta_j p_j$
   8. $p_{j+1}^* := r_{j+1}^* + \beta_j p_j^*$
4. Return x_j und x_j^*

2. Die allgemeinere Aufrufform des Kommandos BICG
   `[x,flag,relres] = BICG(A,b,tol,maxit,M1,M2,x0)`
   akzeptiert analog zu GMRES eine selbstgewählte Startnäherung x_0, lässt Vorkonditionierung mit $M1, M2$ zu und liefert neben der Lösung x ein flag sowie das relative Residuum relres $= \|b - Ax\|_2 / \|Bb\|_2$.
3. A kann auch eine Funktion `afun` bezeichnen, die mit dem Aufruf `afun(x)` den Wert $A * x$ und mit dem Aufruf `afun(x,'transp')` den dualen Wert $A^T x$ liefert.

Beispiel 73 Wir erzeugen das sparse System $Ax = b$ mit unsymmetrischer Bandmatrix A mittels der MATLAB-Kommandos

```
n = 10000; on = ones(n,1);
A = spdiags([-2*on 4*on -on],-1:1,n,n);  A(n,1) = -10;  A(1,n) = 10;
b = sum(A,2);
```

Um die Genauigkeit $tol = 10^{-12}$ zu erreichen, benötigt das BiCG-Verfahren 57 Iterationen. Nimmt man eine Vorkonditionierung mittels der Matrizen

```
M1 = spdiags([on/(-2) on],-1:0,n,n);
M2 = spdiags([4*on -on],0:1,n,n);
[x,flag,relres,iter,resvec] = bicg(A,b,tol,maxit,M1,M2);
```

vor, so werden nur 17 Iterationen gebraucht (vgl. Abb. 20). Der Rechenzeit-Vergleich in Abb. 21 zeigt jedoch, dass der Zusatzaufwand für die Vorkonditionierung hier keinen Zeitvorteil bringt. □

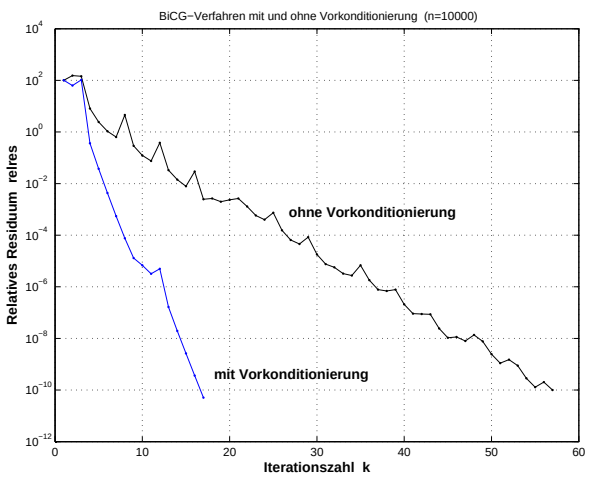


Abbildung 20: Iterationszahl k

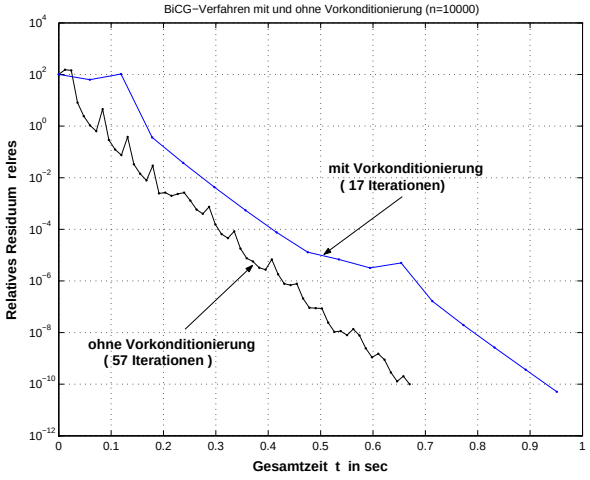


Abbildung 21: Rechenzeit in sec

2.5 Zusammenfassung

Iterative Verfahren approximieren die Lösung großer linearer Gleichungssysteme $Ax = a$ in k Iterationsschriten nur mit moderater Genauigkeit, allerdings mit weit geringerem Aufwand als direkte Verfahren.

1. Als *Splitting-Verfahren* bezeichnet man Iterationsverfahren, die die Koeffizientenmatrix A in die Differenz von $n \times n$-Matrizen aufspalten und nach Umformung ein so genanntes lineares Einschrittverfahren $x_{k+1} = Bx_k + b\,,\ k = 0, 1, 2, \ldots,$ mit Iterationsmatrix B erzeugen. Hinreichend für dessen Konvergenz ist $\|B\| < 1$ mit einer beliebigen Matrixnorm.
2. Das *Jacobi-Verfahren* und das *Gauß-Seidel-Verfahren* sind die bekanntesten Splitting-Verfahren. Während Ersteres den Iterationsvektor x_k stets nach einem Gesamtschritt k überschreibt, wird im Gauß-Seidel-Verfahren der Vektor x_k bereits in jedem „Einzelschritt“ aktualisiert. Starkes Zeilensummen-Kriterium und starkes Spaltensummen-Kriterium sind hinreichend für die Konvergenz beider Verfahren bei beliebigen Startnäherungen.
3. *Diagonal-dominante Matrizen* entstehen häufig bei der Näherungslösung von Randwertproblemen gewöhnlicher und partieller Differenzialgleichungen, z.B. bei der Poisson-Gleichung. Ist A strikt diagonal-dominant oder irreduzibel und diagonal-dominant, so ist A regulär. Gesamt- und Einzelschrittverfahren konvergieren dann stets gegen die Lösung x des linearen Gleichungssystems.
4. *Relaxationsverfahren* entstehen aus Splitting-Verfahren durch Parametrisierung mit einem Relaxationsfaktor $\omega > 0$. Während man bei $\omega < 1$ von Unterrelaxation spricht, bedeutet Überrelaxation mit $\omega > 1$ eine Beschleunigung des jeweiligen Basisverfahrens. Von besonderer Bedeutung sind Jacobi-Überrelaxation (JOR) und Gauß-Seidel-Überrelaxation (SOR), z.B. bei der Vorkonditionierung von Projektionsverfahren.
5. Ein *Projektionsverfahren* bestimmt eine Folge von Näherungslösungen (x_m), $m = 0, 1, 2, \ldots,$ in m-dimensionalen Unterräumen $\mathcal{K}_m$ so, dass eine Orthogonalitätsbedingung erfüllt ist. Galerkin-Bedingungen führen auf orthogonale, Galerkin-Petrov-Bedingungen auf schiefe Projektionsverfahren.

6. *Krylov-Unterräume* $\mathcal{K}_m$ können effizient, also mit wenigen Matrix-Vektor-Multiplikationen, aufgebaut werden. Eine Orthonormalbasis von $\mathcal{K}_m$ liefert das Arnoldi-Verfahren bzw. das modifizierte Arnoldi-Verfahren zusammen mit einer oberen $m \times m$-Hessenberg-Matrix H_m, die die Projektion von A auf den Krylov-Raum repräsentiert.
7. Das *GMRES-Verfahren* ermittelt denjenigen Vektor x_m des Krylov-Raumes, der die Residualnorm minimiert. Restarts nach m Schritten halten den zunehmenden Verfahrensaufwand in Grenzen. Es existieren zahlreiche Varianten und Vorkonditionierungs-Möglichkeiten dieses Standardverfahrens für reguläre Matrizen.
8. Das *bikonjugierte Gradientenverfahren (BiCG)* konstruiert ein Paar biorthogonaler Basen des Krylov-Raumes und des transponierten Krylov-Raumes mittels Lanczos-Biorthogonalisierung. Es benötigt dazu lediglich kurze Dreiterm-Rekursionen.
9. In MATLAB stehen die beiden behandelten und weitere leistungsstarke Krylov-Methoden zur Verfügung, die zusammen mit geeigneten Vorkonditionierern zu den effizientesten Lösern für großdimensionale lineare Systeme gehören. Ausführliche Darstellungen der Vielzahl moderner iterativer Verfahren findet man in der Spezialliteratur [27, 12, 21, 25, 17, 7].

Literatur

[1] Allgower, E. L.; Georg, K.: *Numerical Continuation Methods*, Springer Verlag, Berlin 1990

[2] Carroll, T.; Pecora, L. (Hrsg.): *Nonlinear Dynamics in Circuits. Yoshinaga, T.; Kawakami, H., S.89-120*, World Scientific, Singapore 1995

[3] Ciarlet, P.G.; Lions, J.L. (Hrsg.): *Handbook of Numerical Analysis, Vol. V. Allgower, E.L.; Georg, K.: Numerical Path Following, S. 3-207*, Elsevier, Amsterdam 1997

[4] Ciarlet, P.G.; Lions, J.L. (Hrsg.): *Handbook of Numerical Analysis, Vol. VIII. Van der Vorst, H.A.: Computational Methods for Large Eigenvalue Problems, S.3-179*, Elsevier, Amsterdam 2002

[5] Deuflhard, P.; Hohmann, A.: *Numerische Mathematik I*, 2. Auflage, W. de Gruyter, Berlin 1993

[6] Deuflhard, P.: *Newton Methods for Nonlinear Problems*, Springer Verlag, Berlin 2004

[7] Eisenstat, S.C.; Walker, H.F.: *Choosing the Forcing Terms in an Inexact Newton Method*, SIAM J. Scientific Comput. 17, No.1, S.16-32

[8] Engeln-Müllges, G.; Reutter, F.: *Numerik–Algorithmen mit ANSI C-Programmen*, Wissenschaftsverlag, Mannheim 1993

[9] Golub, G.; Loan, C.V.: *Matrix Computations*, John Hopkins University Press, Baltimore 1989

[10] Gramlich, G.; Werner, W.: *Numerische Mathematik mit MATLAB*, dpunkt.verlag GmBH, Heidelberg 2000

[11] Gustafson, K.E.; Rao, D.K.M.: *Numerical Range. The Field of Values of Linear Operators and Matrices*, Springer Verlag, New York 1997

[12] Hackbusch, W.: *Iterative Lösung großer schwachbesetzter Gleichungssysteme*, B.G.Teubner, Stuttgart 1991

[13] Hanke-Bourgeois, M.: *Grundlagen der Numerischen Mathematik und des Wissenschaftlichen Rechnens*, B.G. Teubner, Stuttgart 2002

[14] Hermann, M.: *Numerische Mathematik*, Oldenbourg Verlag, München 2001

[15] Hoffmann, A.; Marx, B.; Vogt, W.: *Mathematik für Ingernieure. Lineare Algebra, Analysis – Theorie und Numerik*, Pearson Stuidium, München, erscheint 2005

[16] Isaacson, E.; Keller, H. B.: *Analyse numerischer Verfahren.* Verlag Harry Deutsch, Frankfurt 1972.

[17] Kelley, C.T.: *Iterative Methods for Linear and Nonlinear Equations*, SIAM Publications, Philadelphia 1995

[18] Kosmol, P.: *Methoden zur numerischen Behandlung nichtlinearer Gleichungen und Optimierungsaufgaben*, B.G.Teubner, Stuttgart 1993

[19] Latussek, P.; Seidel, H.U.; Vogt, W.; Wagner, E.: *Lehr-und Übungsbuch Mathematik. Band V*, Fachbuchverlag, Leipzig–Köln 1992

[20] Maess, G.: *Vorlesungen über numerische Mathematik. Band 1 und 2*, Akademie – Verlag, Berlin 1984

[21] Meister, A.: *Numerik linearer Gleichungssysteme*, Vieweg–Verlag, Braunschweig 1999

[22] Ortega, J.M.; Rheinboldt, W.C.: *Iterative Solution of Nonlinear Equations in Several Variables.* Academic Press, New York 1970

[23] Philippow, E.S.; Büntig, W.G.: *Analyse nichtlinearer dynamischer Systeme der Elektrotechnik.* Carl Hanser Verlag, München 1992

[24] Piegl, L.; Tiller, W.: *The NURBS Book.* 2nd edition, Springer Verlag, Berlin 1997

[25] Quarteroni, A.; Sacco, R.; Saleri, F.: *Numerische Mathematik. Band 1 und 2*, Springer Verlag, Berlin 2002

[26] Rheinboldt, W.C.: *Methods for Solving Systems of Nonlinear Equations.* 4th ed., SIAM Publications, Philadelphia 1994

[27] Saad, Y.: *Iterative Methods for Sparse Linear Systems*, PWS Publishing Company, Boston 1995

[28] Schwarz, H. R.: *Numerische Mathematik.* B.G. Teubner, Stuttgart 1988

[29] Schwetlick, H.: *Numerische Lösung nichtlinearer Gleichungen.* Deutscher Verlag der Wissenschaften, Berlin 1979

[30] Schwetlick, H.; Kretzschmar, H.: *Numerische Verfahren für Naturwissenschaftler und Ingenieure.* Fachbuchverlag, Leipzig 1991

[31] Stoer, J.: *Numerische Mathematik 1*, 7., neubearbeitete und erweiterte Auflage, Springer Verlag, Berlin 1994

[32] Törnig, W.; Spellucci, P.: *Numerische Mathematik für Ingenieure und Physiker. Band 1 und 2.* 2. Auflage, Springer Verlag, Berlin 1988.

[33] Trefethen, L.N.; Bau, D.: *Numerical Linear Algebra*, SIAM Publications, Philadelphia 1997

[34] Wilkinson, J.H.: *The Algebraic Eigenvalue Problem*, Clarendon Press, Oxford 1992